3D FACE RECOGNITION UNDER VARYING EXPRESSIONS USING AN INTEGRATED MORPHABLE MODEL

SEBASTIEN BENOÎT

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2004

ACKNOWLEDGEMENTS

I would like to thank my supervisor A/P Ashraf Kassim for his guidance throughout the course of this research. I am also very grateful for his help in reviewing this work and improving my command of the English language. I would also like to thank Prof. Y. V. Venkatesh and A/P Terence Sim for the precious advices they gave me at various stages of my research.

My warmest thanks go to Lee Wei Siong and Olivier de Taisne. Our passionate discussions challenged my conceptions and provided me with a fresh perspective on my work. I am very thankful to all my friends who encouraged me and offered their help to create the database of 3D faces, without which the experiments of this thesis could not have been performed.

I would like to sincerely thank my parents for the education they provided and their constant support in my studies and in life in general.

Last but not least, many special thanks to Sylvia for her encouragements, care and support.

TABLE OF CONTENTS

SUMMARY

This thesis introduces a new method to recognize 3D images of faces in a robust manner. It requires no user intervention and applies to the most general type of faces obtained through stereo reconstruction. We describe a novel approach, using an "Integrated Morphable Model" (IMM), which improves on the "morphable model" framework to recognize faces under varying expressions.

IMMs are created using a symmetric matching scheme for computing correspondences between examples faces, which yields more accurate results than earlier algorithms. Submodels are computed for each person in the database and merged to form a IMM that takes into account both intra-personal and extra-personal variations in our database. Recognition is performed by morphing the model to an arbitrary input face and classifying the input using the morphing parameters.

We present experimental results showing good recognition rates, and confirming the validity of our approach.

LIST OF SYMBOLS

$\vec{p}$	A 6D vector p.
$\mathcal{A}$	Mesh $\mathcal{A}$.
$\widehat{\mathcal{M}}$	A morphable model $\mathcal{M}$.
$\vec{P}_{\mathcal{M}}(\vec{q})$	Projection operator yielding the point of surface $\mathcal{M}$ closest to $\vec{q}$.
$\mathbf{C}$	Correspondence field $\mathbf{C}$: a set of displacement vector.
$\mathbf{C}(\vec{p})$	Returns the corresponding point to $\vec{p}$ using $\mathbf{C}$.
$\mathbf{C}$ ($\mathcal{A}$)	Returns the corresponding mesh to $\mathcal{A}$ using $\mathbf{C}$ on all vertices of $\mathcal{A}$

.

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

Chapter 1

Introduction

Face recognition is one of the most significant application of pattern recognition and computer vision and has attracted considerable attention over the last three decades. While humans excel at recognizing faces, it remains a daunting challenge for machines, especially from intensity images. However, if tackled in three dimensions the face recognition problem is unquestionably more tractable. With 3D stereo capture devices becoming commonplace and research on stereo reconstruction steadily improving (see [25] for a survey of the domain), there is a growing need for robust algorithms making full use of the 3D geometry to achieve better recognition results. This thesis aims at designing one such algorithm.

1.1 The 3D Face Recognition Problem

1.1.1 The General Face Recognition Problem

Definition Formally stated, a face recognition problem involves identifying 2D, 3D or video data of a human face using a stored database of faces. We have to further distinguish between face recognition and verification[1]. In a recognition problem the input face is unknown and we attempt to determine if it belongs to the database and to which person. In a verification problem we have to verify if

[1]Recognition as defined here is also sometimes called a "watch list problem" to further distinguish it from the identification problem in a closed universe (i.e., we know the input belongs to the model, and we only seek to identify it).

two given faces belong to the same individual, or if a given face belongs to a group of individuals. Verification can be considered a subset of recognition (one to one instead of one to many recognition), hence we will tackle the more general case of recognition.

Related problems In this thesis we are not concerned with face tracking, a related problem which involves the localization of human faces in a scene, but it could be a front-end to our face recognition system.

Another set of related problems is the analysis of human expressions, emotion and/or expression recognition. While we have not conducted experiments in this field our algorithm could be adapted for emotion recognition and we briefly describe an approach to serve this purpose in chapter 6.

1.1.2 3D vs. 2D Face Recognition

After years of face recognition research based on 2D images, there is no universally accepted solution that gives satisfactory results outside of a controlled environment. 3D images in contrast are generally believed to have "the potential for greater recognition accuracy than the use of 2D face images" [8]. Indeed the shape information of 3D data is not sensitive to pose or illumination and appears better suited to describe a face, which is essentially a 3D object, than 2D intensity images.

This assumption was recently verified by Chang et al. in [6]. By applying the same PCA-based method to both 3D and 2D data on a large dataset they found that the 3D-based system consistently outperformed the 2D-based system.

1.1.3 Recognizing Faces Under Varying Expressions

The major challenges of a face recognition system have long been identified the variations of pose, illumination or expression [34, 8]. In the case of 3D face recognition, the two first problems are less important [8]. Indeed illumination changes have no effect on the geometry of the mesh and the pose problem becomes one of recovering the 3D alignment of faces, which has been studied already [28]. In any case it is considerably easier than in 2D where we have to recover *lost* information due to pose variations.

But face recognition under varying expressions remains a challenging problem, as pointed out by Chang et al. [8]. Most algorithms report results on datasets that contain no or very limited expression changes in the test subjects. It has been shown performance drops dramatically for 3D PCA-based systems [8].

Therefore, in this thesis, we focused on recognition of 3D faces under varying expressions.

1.1.4 Applications of 3D Face Recognition

Key applications for face recognition were presented in a general survey in [34]. Some of these applications are reproduced in table 1.1 below.

One may argue that not all of these applications can realistically use 3D input faces, at least in the immediate future. This is especially true for surveillance applications, for instance video surveillance. While there has been tremendous research efforts to recover structure from motion, current algorithms lack accuracy and robustness [21] and therefore cannot be directly used for 3D face recognition today.

However even excluding this category of applications, we can see from table 1.1 that

Area	Application Examples
Biometrics	Drivers Licenses, Entitlement Programs National ID, Passports, Voter Registrations Welfare Fraud
Information Security	Desktop Logon Application Security, File encryption Network Security
Smart Cards	Stored Value Security, User Authentication
Access Control	Facility Access, Vehicular Access
Law Enforcement and Surveillance	Advanced Video Surveillance Portal, Post-Event Analysis Shoplifting and Suspect Tracking

Table 1.1: Applications of face recognition

out of 19 applications listed in this table 14 are compatible with 3D-based recognition. In these areas (biometrics, access control, information security) current stereo capture devices can be installed easily. Since the capture itself is instantaneous (the time of a snapshot) we are limited only by the processing time of the reconstruction algorithm and this figure would drop as the computational power of computers increase. Hence a wide range of applications could hugely benefit from 3D face recognition as this type of device become cheaper and more readily available.

1.2 Thesis Overview

Our aim is to design a complete framework for the recognition of 3D faces which is able to cope with variations due to expression changes without any user intervention.

We represent faces with arbitrary triangle meshes since this is the most commonly used representation for 3D data. The vertices of the meshes are 6D vectors,

since we use 3 coordinates for representing the geometry and 3 coordinates for the colors values (red, green and blue). We assume that the surfaces have no "holes", as this is consistent with the output of most 3D capture devices. With these assumptions, the face recognition problem becomes one of recognizing a surface embedded in a 6D vector space.

1.2.1 Overview of our Approach

Our algorithm is divided in two stages as summarized in figures 1.1 and 1.2.

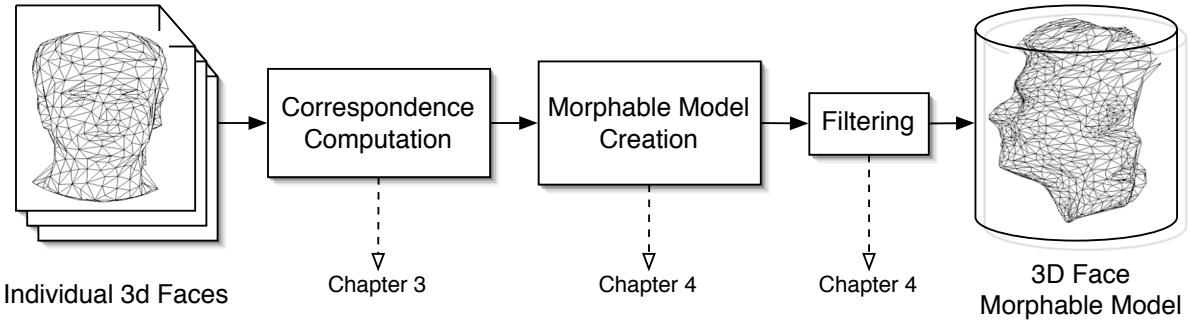


Figure 1.1: 3D Face model creation stage (performed offline).

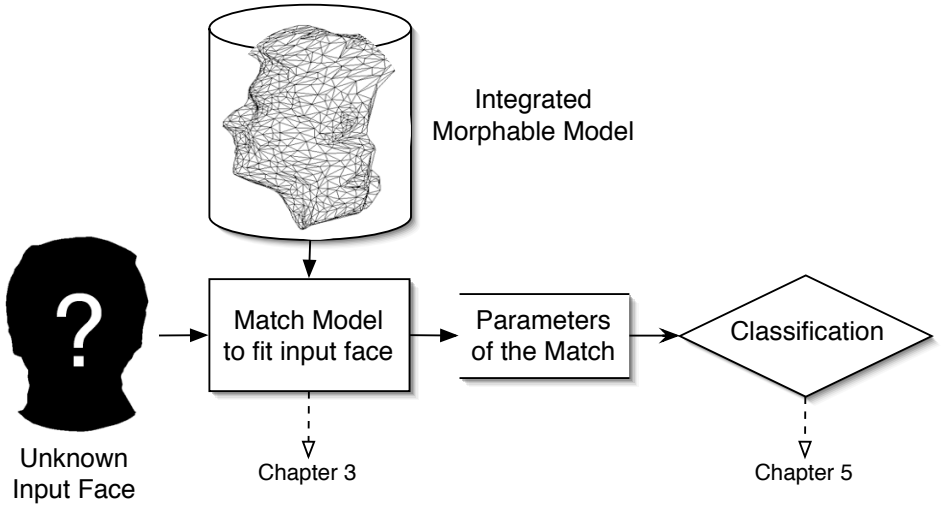


Figure 1.2: 3D face recognition stage

The first stage is performed offline and is the most computationally intensive. We create a morphable model out of faces of the persons to be recognized. Morphable models are originally described in [30]. The morphable model encodes the class-specific information of the faces of a training set and can be deformed to match any arbitrary mesh. Prior to building a morphable model, a key step is computing correspondence fields between surfaces, detailed in chapter 3.

Once the correspondences are found for all vertices of the meshes, the morphable model consists in: a base surface, which is the computed average of the input meshes, and sets of displacement vectors, which are the correspondence fields between the input mesh and the base. We further improve on the basic morphable model by integrating submodels of individuals to capture both intra- and extra personal relations, and we refer to the resulting model as an *Integrated Morphable Model* (IMM). The process is described in chapter 4.

The capture process sometimes results in random artifacts on the surface mesh. To make the model more robust, we selectively apply a filter based on curvature operators to the input meshes to reduce those artifacts (chapter 5).

The recognition stage is summarized in figure 1.2. The model described above is deformed to fit an input face and the parameters are used to classify the face. We describe our classifier in chapter 5 and we present experimental results for recognition purposes.

1.2.2 Contributions Summary

We summarize below the key contributions of this work:

- A novel 3D face recognition system applicable to any type of surfaces (without holes) with little or no preprocessing.

- Improved automatic 3D correspondences computations: more accurate symmetric correspondences using adapted data structure for more efficiency.
- Introduced selective filtering to eliminate artifacts from the capture device, making the morphable model creation more robust.
- To a lesser extent, we also applied our IMM to *expression synthesis* to demonstrate its capacity to capture human expression variations (in chapter 5).

Chapter 2

Related Works

There has been a considerable amount of work in face recognition in the past 30 years. A survey of these can be found in [34, 8]. These works can be categorized according to the dimension of their input data: either 2D (intensity images), or 3D (surfaces, meshes or range data).

2.1 2D Model-Based Face Recognition

While 2D face recognition systems perform reasonably well in a controlled environment (frontal, expressionless views with controlled lighting) they fail to give satisfactory results whenever the pose, illumination or facial expression changes. In order to be able to handle variations of a face's appearance, we have to incorporate information about the class the face belongs to. This class-specific knowledge can take the form of a model of allowable deformations of a face. This was first suggested less than a decade ago in [4] and [32], at a time when direct recognition from 3d object was not yet contemplated because of the cost and availability 3d capture devices.

Since then many different approaches have been proposed to make use of *a priori* knowledge to recognize faces under varying conditions. Among them, the *morphable model* approach of [30], primarily intended for the synthesis of faces, is one of the most successful. A morphable model contains examples of a class of objects set in correspondence so that linear combinations of these examples

generate arbitrary new objects belonging to the same class. We use a morphable model representation similar to [27] where the morphable models can be extended to any surface embedded in a nD-dimensional space although it was not applied to face recognition. However, our method for constructing the morphable model differs in that we use a symmetric scheme to achieve closer approximation to the faces (detailed in chapter 3), and we obtain a global morphable model by merging sub-models of every individual (explained in chapter 4).

Comparison of Similar Works with our Approach

Face recognition using morphable models have been attempted in a few papers, most recently in [5], a continuation of [30]. A linear Support Vector machines is used for the classification and trained with varying illumination and pose. They report good results for verification (about 95%) but up to 8 corresponding feature points have to be manually labelled in each image. Moreover, test subjects with varying expressions were discarded to obtain these results. In contrast, our algorithm is fully automatized and has been tested for face recognition, which is more general that face verification with data sets where the subjects have very different expressions in the training and tests sets.

Another method [16], combines component-based recognition and morphable models constructed with the same technique but uses the same algorithm as [5] for creating morphable models and hence suffer from the same shortcomings.

2.2 3D Face Recognition

As indicated in the introduction, few contributions were made using 3D faces as direct input, primarily because of the cost and limited accuracy of the early 3D face scanners.

According to a recent survey of 3D face recognition (see [7]), of all algorithms attempting to recognize 3D faces, emotion recognition is a key challenge that is seldom addressed by current algorithms:

> *Approaches that effectively assume that the face is a rigid object will not be able to handle expression change.[. . .]Clearly, variation in facial expression is a major cause of degradation that must be addressed in the next generation of algorithm.*

In the following, we broadly categorize works in 3D face recognition according to the type of 3D data used, their recognition approaches and wether they combine 2D and 3D data to recognize a face.

2.2.1 Different Types of 3D Data: 3D vs 2.5D

There are mostly three types of methods for capturing 3D data:

- *range scanners*; where a laser is used to measure the distance to each point of the face and outputs a distance map.
- *structured light*; where a pattern is projected on the face and the analysis of the deformation of the pattern provides information about the 3D structure.
- *stereo reconstruction*; where a 3D mesh of the face reconstructed by combining information from multiple high-resolution cameras.

These three techniques are not equivalent: only *stereo reconstruction* can extract the surface texture and output data in the most general type of representation (triangle meshes). In comparison, range images offer an an implicit data representation –also called 2.5D or *depth map* – and structured light yields very sparse measurements.

Most works [12, 10, 29] use range data obtained with laser scanners which are slower than stereo capture and do not extract the surface texture. Hence they often rely on an implicit data representation and their method cannot be applied to more general triangles meshes – in full 3D– like those output by a stereo system. Others, like Beumier [3], used structured light but as noted in a recent study of capture technology (see [20]) current structured light approaches are not accurate enough for face recognition[1].

In contrast our technique is based on general triangle meshes and therefore is largely device-independent. Moreover the stereo reconstruction is faster than conventional range scanners. We also provide a technique to remove artifacts produced by the capture device.

2.2.2 Using Curvature or PCA for Recognition

The vast majority of studies use curvature properties to uniquely identify the faces since they are invariant to the viewpoint and illumination. Gordon [12] made one of the first successful attempts to recognize faces from ranges images, by extracting a set of features and comparing their relative measurements. He reported very high recognition results but noted that the computed features are

[1]Instead they suggest a combination of stereo and structured light reconstruction might lead to better results.

usually similar for a given face *"except for the case with large feature detection error or variation due to expression"*, which suggest that the algorithm cannot handle changes in expression.

Tanaka et al. [29] used more complex feature vector based on an Extended Gaussian Image (EGI) to uniquely describe a point's local properties. They report good result on dataset containing no expression variations.

Moreno et al. [19] segment the face using the mean and Gaussian curvatures and extract a set of 35 features from the segmented meshes. They tested their system on a dataset containing pose and some expression variation and obtained between 71% to 78% for overall recognition, the top score is reached when all features are visible in the images. They used general 3D images obtained by stereo reconstruction and studied the effect of expression changes: results drop to 45% to 62% when the expression varies (and even lower when all features are not all visible). However they did not try to specifically model the deformations due to expression changes but instead chose features (near the nose especially, the eyes...) that will not be dramatically affected by a change of expression. An obvious drawback of their technique is that the results depend on finding the appropriate features, which is not always feasible in noisy images. We compare our results against theirs in section 5.3 since it is one of the rare few works including some expression variations in their dataset.

Our approach does not rely on the local curvature properties of 3D surface, except for the (optional) filtering in chapter 5. Indeed there is no universally accepted definition of curvature for triangle meshes (because they are discrete, whereas for continuous surfaces there is a unique definition) and in our experience the curvature (mean or Gaussian) is quite sensitive to noise.

Other works have tried to apply methods that were successful for 2D images to 3D by taking advantage of the 2.5D representation of range images; Hesher et al. [13], for instance, extended Principal Component Analysis (PCA) for range images using 6 images with different expressions in their training sets. However they do not specify if their test set have varying expressions (the survey by Chang assumes they do not [7]) Moreover their method relies heavily on the specific representation of range images.

Achermann et al [2] uses the Hausdorff distance (a minimax function) to classify general 3D faces but he considered them as 3D points clouds rather than 3D surfaces.

The approaches discussed in this section treated face recognition as a problem of fitting rigid surfaces and therefore could not account for non-rigid deformation that occur when subjects exhibit different expressions.

2.2.3 Multi-Modal Methods

Recently, more attention has been given to multi-modal methods which combine different data sources to recognize faces (most commonly 3D and 2D[2]).

Wang et al. combined Gabor filter responses in the 2D domain and Point Signature in 3D (a feature vector similar to Tanaka et al. in [29]). They use a Support Vector Machine to classify these outputs and report 70 to 90% recognition. Their database include different expressions for their subjects but they require many feature points to be selected in all faces in 2D and 3D.

Bronstein et al. in [9] has attempted expression-invariant face recognition.

[2]The use of IR images , iris scans, the gait, voice data. . . have also been proposed but are not covered here.

They assume that the transformations undergone by a face are always isometric and combine the "bending invariant canonical form" of the 3D geometry (from a range scanner) and a flattened texture image. Recognition is performed by using a variant of the Eigenfaces method. This method is probably the first to take into account that the faces are deformable (some manual intervention required). Unfortunately their results are not reported.

Chang et al. [6] adapted an eigenface decomposition on a fusion of 3D range images and 2D snapshot. They reported 94% recognition when using 3D face recognition alone and more than 98% using the fusion of 2D and 3D, with no expression variations in their test subject. In a later work (see [8]) they described experiments to study the effect of expression change on their algorithm, and reported a performance drop to 55% for 3D face recognition.

2.2.4 Summary

While most papers presented in this section merely attempt to match rigid facial surfaces, our representation based on Hierarchical Morphable Models is capable of deforming a base 3D surface to fit a particular expression. Our algorithm has been tested with test subjects showing very different expressions and gave satisfactory results without requiring manual intervention.

We compare our results with those of the two only other works tackling expression changes; i.e., by Moreno et al. [19] and Chang et. al [6].

Chapter 3

Surface Correspondences

3.1 Introduction

Problem definition. In order to build a *morphable model* we need to compute correspondences between the 3D surfaces representing the faces. For all points on a "source" mesh, we want to find the corresponding point on a "target" mesh. By corresponding point we mean that, for instance, a point on the tip of the nose of the first face should correspond to the tip of the nose in the second face, a point on the left earlobe should correspond to a point on the left earlobe and so on. We can only give this informal definition since the correspondences are not uniquely defined for each face. Each correspondence can be defined by a 6D *displacement vector*: 3 euclidean coordinates (x, y, z) and 3 color values (r, g, b); which is simply the difference between a point in the source mesh and its computed corresponding point in another mesh. The set of displacement vectors forms a *displacement field*, which when added to the source mesh gives an approximation of the target mesh. If we add only a fraction k (with $k \in [0, 1]$) of this correspondence field we obtain an intermediate mesh between the source and the target, whose resemblance to the target mesh increases with k.

This is summarized in equation 3.1 below.

$$\mathcal{A}' = \mathcal{A} + k \cdot \mathbf{C}_{\mathcal{B}}(\mathcal{A}),\ k \in [0, 1] \tag{3.1}$$

where $\mathbf{C}_{\mathcal{B}}(\mathcal{A})$ is the displacement field between mesh $\mathcal{A}$ and mesh $\mathcal{B}$ (with the

addition symbol defined over the vertices of $\mathcal{A}$).

Morphing. *Morphing* is the operation of gradually augmenting the coefficient k in equation 3.1, to obtain a sequence of images showing a smooth transition between two different faces. We can judge the quality of the computed correspondences by the quality or *realism* of the morphing and by the closeness of the final approximation to the target mesh. An example of morphing is presented in figure 3.1

Figure 3.1: A typical morph sequence between two faces. It was computed with our symmetric matching algorithm.

Previous Works. The computation of correspondences is not a well-defined problem because they are not unique. Faces are deformable surfaces and, in contrast to rigid bodies like CAD models whose correspondences have been extensively studied for recognition and retrieval, there is no closed form solutions and few attempts at solving this problem. In their first morphable model description, [30], Blanz and Vetter used a modified optical flow algorithm, initially meant for gray-level images that they adapted to cylindrical coordinates. However this algorithm cannot be used with full 3D meshes which are more general than the surfaces described with implicit representations – in which one coordinate represents depth.

Shelton did excellent work in [26, 27] describing a correspondence algorithm capable of matching any surface embedded in a nD-dimensional space and how to use them to build morphable models. We adapted his work, and made two major improvements: we reduced the complexity of the algorithm, initially very high, with a heuristic valid for smooth surfaces like faces, and we improved the quality of the correspondences by using a symmetric matching scheme.

3.2 Computation of Surface Correspondences

We summarize here a theoretical framework for the computation of correspondences based on Shelton [27] and Hoppe [15] . A more rigorous approach can be found in Hoppe's work [15, 14], whose formulation is very close to Shelton's, albeit with the goal of mesh optimization.

Definitions. Let $\mathcal{A}$ be our "source" mesh and $\mathcal{B}$ our "target" mesh. A triangle mesh can be visualized as a piecewise linear surface composed of triangle patches pasted together along their edges. It is more formally defined as a set of vertices: the 6D vectors (position and color) and their connectivity: pairs of vertices constitute the edges and triplets of vertices constitute the faces. Our problem can be stated as: for each vertex in $\mathcal{A}$ we wish to find the corresponding *point* in $\mathcal{B}$ – it is not necessarily a vertex of $\mathcal{B}$.

3.2.1 Problem Statement: Minimizing an Energy Function

Motivations. To solve this problem we can attempt to minimize a distance function between the two meshes, by iteratively displacing the vertices of $\mathcal{A}$ in order to match $\mathcal{B}$ as closely as possible, and the final approximation of $\mathcal{B}$ will give

us the correspondences – the difference between the position of $\mathcal{A}$'s vertices after all iterations and their original positions. The definition of this distance function is crucial for the quality of the correspondences. If we simply define the geometric distance between the two mesh then we only achieve a global *projection* of mesh $\mathcal{A}$ onto mesh $\mathcal{B}$, each vertex of $\mathcal{A}$ will move anywhere on $\mathcal{B}$ as long as it is the *geometrically* closest possible match. As a result, the correspondences are wrong (nose match to eye and so on..), and the intermediate meshes look distorted. We need to incorporate some knowledge about the structure of the mesh, to penalize vertex moves that result in distorting the structure.

Energy function. To achieve this Hoppe suggested to minimize an *energy function* [15], which was further improved by Shelton [27]. We used Shelton's definition below:

$$E(\mathbf{C}) = E_{similarity}(\mathbf{C}) + \alpha \cdot E_{structure}(\mathbf{C}) + \beta \cdot E_{smoothness}(\mathbf{C}) \tag{3.2}$$

where $\mathbf{C}$ is the *correspondence function*, i.e. a function mapping a point $\vec{a}$ on mesh $\mathcal{A}$ to any point in space $C(\vec{a})$. We will minimize E with respect to $\mathbf{C}$ iteratively refining the correspondences at each step. We can consider $\mathbf{C}$ as an "update" of the positions of the vertices of mesh $\mathcal{A}$. At each round of iteration the vertices of $\mathcal{A}$ are moved according to the solution function $\mathbf{C}$ thus obtaining an approximation $C(A)$ that will be used for the next iteration and so forth.

The different energy terms in equation 3.2 ensure that $\mathbf{C}$ is a "good" correspondence as described above intuitively. $E_{similarity}$ is the Euclidean distance between the two meshes and therefore ensures that the approximation $C(A)$ will be as close to $\mathcal{B}$ as possible, while $E_{structure}$ aims at preserving the structure, penalizing "bad" correspondences that distort the mesh. The last term, $E_{smoothness}$ is a

regularization parameter, to ensure the approximation $C(A)$ is smooth and to discourage jagged meshes. Finally, the coefficients α and β give a trade-off between the relative importance of the *similarity*, *structure* and *smoothness* energy terms.

3.2.2 Practical Instantiation of the Energy Function

In what follows we define the terms of equation 3.2.

Similarity term. $E_{similarity}$ is simply the Euclidean distance between the two meshes, it can be defined as the sums of the distances between each mesh' vertices and its closest point on the other mesh:

$$E_{similarity}(\mathbf{C}) = \int_{\mathcal{A}} \|\vec{a} - \vec{P}_{\mathcal{B}}(\vec{a})\|^2 d\vec{a} + \int_{\mathcal{B}} \|\vec{b} - \vec{P}_{\mathcal{A}}(\vec{b})\|^2 d\vec{b} \tag{3.3}$$

where $\vec{a}$ and $\vec{b}$ are 6D vectors, $\vec{P}_{\mathcal{M}}(\vec{p})$ is an operator yielding the point of surface $\mathcal{M}$ closest to $\vec{p}$, i.e., the projection of $\vec{p}$ on $\mathcal{M}$.

Structure term. There are many ways to characterize the structural properties of a mesh, for instance using the local curvature or the bending energy of a surface. Hoppe [15] suggests the use of a network of springs spread over the mesh which can be easily computed and Shelton [27] improved his formulation by using *directional* springs instead. Directional springs try to preserve their original length as well as their orientation, as detailed in [26]. This leads to the following definition for $E_{structure}$, the total energy of a network of directional springs:

$$E_{structure}(\mathbf{C}) = \int_{\mathcal{A}} E_{ds}(\vec{a}, \vec{a} + d\vec{a}, \mathbf{C}) d\vec{a} \tag{3.4}$$

with $E_{ds}(\vec{p}, \vec{q}, \mathbf{C})$ the energy of a directional spring connecting $\vec{p}$ and $\vec{q}$, two points on the mesh $\mathcal{A}$:

$$E_{ds}(\vec{p}, \vec{q}, \vec{c}_{\mathcal{B}}) = \frac{\|(\vec{p} - \vec{q}) - (\mathbf{C}(\vec{p}) - \mathbf{C}(\vec{q}))\|^2}{\|\vec{p} - \vec{q}\|} \tag{3.5}$$

In optimization terms, $E_{structure}$ performs a regularization that helps guide the minimization to a desirable local minimum. ([15]).

Smoothness term. To penalize discontinuous surface, we use a similar spring network, but composed of a normal spring of rest length 0:

$$E_{smoothness}(\mathbf{C}) = \int_{\mathcal{A}} E_s(\vec{a}, \vec{a} + d\vec{a}, \mathbf{C}) d\vec{a} \tag{3.6}$$

with $E_s(\vec{p}, \vec{q}, \vec{C})$ the energy of a spring connecting $\vec{p}$ and $\vec{q}$, two points on the mesh $\mathcal{A}$:

$$E_s(\vec{p}, \vec{q}, \vec{c}_{\mathcal{B}}) = \frac{\|\mathbf{C}(\vec{p}) - \mathbf{C}(\vec{q})\|^2}{\|\vec{p} - \vec{q}\|} \tag{3.7}$$

$E_{smoothness}$ is a classic regularization parameter that penalizes unsmooth results.

Trade-off coefficients. The coefficient β is easy to set since the effects on the final result are limited, it can be chosen according to the desired level of smoothness for the final approximation. α is much more important since it determines the strength of the structure preserving term. With a high α value the mesh' vertices will hardly move since any small move might distort the mesh and result in high penalty, but if α is too low the structure term will have no effect and the approximation will simply be a projection of the source $\mathcal{A}$ on the target $\mathcal{B}$.

3.2.3 Solving the Minimization

All the above equations use integrals that we can approximate by sampling points on the surfaces to make them tractable, which amounts to replacing all the integral

symbols in the above equations by discrete sums over the sampled points.

Algorithm 3.2.1: Computation of correspondences

Data: Source Mesh $\mathcal{A}$ and target Mesh $\mathcal{B}$
Result: Correspondences function **C**

Initialize the structure coeff.: $\alpha \leftarrow \alpha_0$
Initialize the current approximation of $\mathcal{B}$: $\mathcal{A}' \leftarrow \mathcal{A}$
repeat
  repeat
    ProjectedPoints $\leftarrow$ `FindProjections`$(\mathcal{A}', \mathcal{B})$
    `ComputeEnergy`$(E, \alpha,$ ProjectedPoints$)$ using equation 3.2
    `Minimize`$(E, \alpha,$ ProjectedPoints$)$ /*s.t. constant projections */
    `UpdateMesh`$(\mathcal{A}')$
  until *no more change in* A'
  `DecreaseStructureCoeff`(α) /* $\alpha \leftarrow \alpha * AnnealingRate$ */
until E *is below a threshold*
$C \leftarrow$ `GetDisplacements`$(\mathcal{A}, \mathcal{A}')$

We can obtain the correspondences by combining two iterations (see algorithm 3.2.1):

(a) An outer loop that refines the correspondences by gradually decreasing the structure term α.

(b) An inner loop solves the minimization of equation 3.2 for a given α.

The outer loop consists simply of multiplying α by an annealing factor. As for the inner loop, we observe that the projection operator $\vec{P}_M(\vec{p})$ in equation 3.3 (called `FindProjection` in algorithm 3.2.1) giving the closest point to a mesh is not linear so we cannot solve 3.2 directly. But if held constant, all the remaining terms are quadratic so we can use the classic least mean square solution to minimize 3.2. Thus we solve the minimization problem iteratively, keeping the projections constant at each iteration and using the updated mesh to recompute new projections and so forth. The detailed implementation is described in [27].

Fast standard inversion techniques like conjugate gradient (from [23] for instance) can be applied to solve 3.2 very efficiently.

The complete algorithm, summarized in 3.2.1, gives reasonably good correspondence fields without any user intervention. However there is still room for improvements, which is disussed in the following section.

3.3 Complexity Improvements

The algorithm presented by Shelton in [27] is very slow despite the use of efficient minimization techniques. It takes about an hour to complete on an Apple G4 Dual 1.4GHz for a resolution of 1200 vertices, and about 9 hours when we double the resolution.

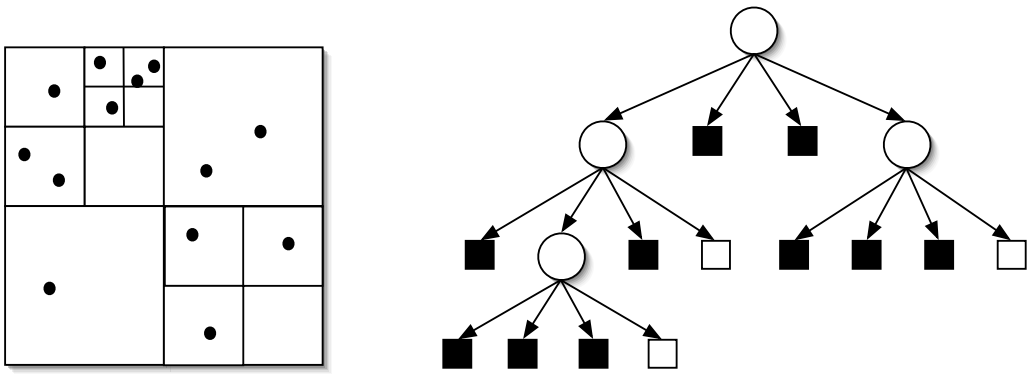
One of most computationally expensive operation is the function $\vec{P}_{\mathcal{M}}(\vec{p})$ (labelled "FindProjection" in algorithm 3.2.1) which returns the point of mesh $\mathcal{M}$ closest to $\vec{p}$. It requires near-exhaustive search on the mesh $\mathcal{M}$ and it is called at every iteration since it is needed to compute $E_{similarity}$ defined in 3.3. Therefore a reduction of the time complexity of this function will directly improve the overall complexity of the algorithm.

In the following subsection we first describe the initial data structure used to perform search queries on the mesh. We then present two methods to reduce the complexity of this search function. In sub-section 3.3.2 we describe an improvement of the data structure. Lastly in sub-section3.3.3 the second method introduces an approximation to further accelerate the search.

3.3.1 The Initial Data Structure

A naive way of computing $\vec{P}_{\mathcal{M}}(\vec{p})$ would be to search extensively all points on the mesh $\mathcal{M}$ for the one closest to $\vec{p}$, and this would imply a complexity proportional to the number of points we sample on the surface, which is unacceptable. Fortunately we can use a better data structure to speed up the search and avoid parts of the mesh that are too far from $\vec{p}$. Possible solutions include *quadtrees*, *octrees*, *k-d trees*, etc. (refer to [24, 33] for a detailed description).

Quadtrees store data in a recursive way so that retrieval takes only $O(\sqrt{N})$ (in 2D) where N is the number of elements in the mesh. A general quadtree is a rooted tree whose elements are vectors of a d-dimensional space. In 2D, it is created by recursively dividing the space into squares. When a square contains too many points (this can be fixed with a threshold), we divide it into four (2^2) squares and a new subtree (with four branches) is created. Figure 3.2 gives a classical representation of quadtrees.



(a) (b)

Figure 3.2: An example of quadtree. Figure(a) shows a region of space recursively divided into squares containing points. Figure(b) shows the resulting quadtree.

In 3D, the quadtree is referred to as an *octree* and the spaces is divided into

"boxes" instead of squares. A general d-dimensional quadtree is a tree with 2^d branches at each level and implies division of the d-dimensional space into 2^d boxes. Search is performed by finding the closest box recursively until only a few points are left in the innermost box, which avoids many computations.

Shelton's original algorithm is based on a 6-dimensional quadtree, which is consistent with dimension of the vector space. This implies having $2^6 = 64$ branches at each level, or 64 "boxes".

Elements of the quadtree. Solving the minimization problem defined in section 3.2 requires finding the closest point on the mesh, but naturally we cannot store *all* points of the mesh in the quadtree. Sampling points on the surface would be slow, as we would have to sample 5 to 10 times the number of vertices to obtain a good approximate.

Instead Shelton implemented a quadtree structure which stores the triangles [26] (indeed quadtrees are not limited to points and can store other geometrical objects, see [24] for a survey). Thus the closest point on the mesh is found by first searching for the closest triangle, then finding the closest point on the resulting triangle.

The two improvements we bring in the following sub-sections concern the dimension of the quadtree and the elements it contains.

3.3.2 Improving the Data Structure

The original algorithm by Shelton uses a 6-dimensional quadtree, which amounts to dividing the space in 64 "boxes". A careful study of the distribution of vectors in the quadtree shows that only a few of these "boxes" are full, while most are

empty. This stems from the different nature of the 6 dimensions in our vector space: the 3 first represent geometry and the 3 last represent color values. The color attribute of a human face does not take up the entire available spectrum (i.e $[0, 256]^3$). For instance green or blue shades are rare (except near the eyes) and most of the face has color values close to the skin tone or the color of the hair. Hence it does not make sense to index the quadtree with color information; since most points have the same color they will be stored in the same box, and many other boxes on the same level will be empty[1].

Therefore it is faster to search a 3D space rather than a 6D space, while keeping the vectors in a 6D representation. This is confirmed by two other well-known facts about hierarchical data structures: a tree is best utilized when it is homogeneous (all levels are similarly occupied), and their efficiency diminish exponentially with their dimensionality – this is known as *the dimensionality curse.*

Using octrees. We implemented this search involving only the 3D space using an octree instead of the 64-dimensional quadtree. Thus it divides the 3D space into $2^3 = 8$ boxes instead of 64 previously. The elements of the hash are still 6D vectors, but when we build the tree we use only the geometrical coordinates (3D) to decide in which box (or node of the tree) we place the data.

According to [24], the worst-case complexity of a d-dimensional quadtree is given by:

$$C(N) = O(d \cdot N^{1-\frac{1}{d}}) \tag{3.8}$$

Therefore by using an octree instead of a 6D quadtree we reduce the complexity

[1]Other data structurelike k-d trees would be more suited since they can locally adapt to the data (i.e. change the size of the "boxes" whereas for a quadtree they are all identical)

from $O(6{\cdot}N^{\frac{5}{6}})$ to $O(3{\cdot}N^{\frac{2}{3}})$. Another consequence is that the octree is more *balanced* (fewer boxes are empty) than the previous structure and the average complexity should also improve [24].

We observed up to **55% speed improvement** when using an octree-based search function. This also accelerated the overall algorithm for correspondence computation. It took only about 30 minutes for a resolution of 1200 vertices and 5 hours for the double resolution instead of respectively 1 and 9 hours needed previously.

3.3.3 Approximate Search with an Edge-Based Heuristic

Let n be the number of vertices in a given mesh. As seen in section 3.3.1, Shelton used a quadtree containing the triangle information to avoid using a quadtree of sampled points (which would require $5n$ to $10n$ elements in the quadree). Nevertheless there are still approximately three times more triangles in a mesh than vertices, so the complexity is $O(9 \cdot n^{\frac{2}{3}})$ (using the formula of the previous section).

We would like to further reduce this complexity to $O(3 \cdot n^{\frac{2}{3}})$ by using only a quadtree containing the vertices. Since we need to find the closest point on the mesh and not only the closest vertex, the result will be approximate. We introduce a method to ensure that our approximation is always within acceptable limits.

Approximating the search. To control the validity of our approximation, we computed the relative euclidean distance between the approximate result using the vertex-based approximation and the triangle-based exact result. It gives a measure of the relative error caused by our approximations.

We observed that if we directly approximate the closest point with the closest

vertex, the approximation will be very far from the exact result in the regions where the triangle have large areas and the error will be very high.

We can refine this approximation by searching for the closest point in a neighbourhood (i.e. in the adjacent triangles) of the five closest vertices. This technique diminishes the average error but it still leads to 5% average error and even 200% in some regions of the mesh, which is unacceptable.

Edge-based heuristic. Therefore we need a criterium to decide whether or not to use our approximation, and when the approximation is valid. We found that comparing the maximum edge length in neighbourhood of the closest vertex to the average edge length in the entire mesh is a good indicator of the validity of our approximation. Intuitively, if the edges are locally larger than in other portions of the mesh, then the vertices are locally farther apart than in most of the mesh and the closest vertex will be a very bad approximation to the closest point. This is true if most of the mesh is "well-behaved" or continuous enough. This heuristic might not work well for very spurious datasets or too complex shapes, where the edge length can vary a lot from the average edge length.

Thus, we use two quadtrees, one for the vertices and the other for the triangles. Depending on the criteria described above, we compute an approximation to the closest point using the vertex-based quadtree (complexity $O(3 \cdot n^{\frac{2}{3}})$) or we compute the exact solution with the triangle based quadtree (complexity $O(9 \cdot n^{\frac{2}{3}})$). The algorithm is summarized in 3.3.1.

However this algorithm implies using two quadtrees instead of one, so the time complexity reduction is compensated by higher space complexity.

Results. We verified experimentally that with this heurisitc the error is always below 0.1% [2].

Using this heuristic we achieved up to 20% speed improvement. If we combine the speed-up of this section and the improvements of the previous section we obtain an average of 65% improvement. The correspondence algorithm using the combined speed-ups took less than 20 minutes for a resolution of 1200 vertices and 3.5 hours for the double resolution (as compared to respectively 1 and 9 hours previously)

Algorithm 3.3.1: Approximate search for the closest point on a Mesh

Data: A Mesh $\mathcal{M}$ and a point $\vec{p}$
Result: $\vec{p'}_{\mathcal{M}} = \vec{P}_{\mathcal{M}}(\vec{p})$

Find the 5 closest vertices searching a hash table of vertices
Compute maxedge
if *maxedgelength* < *meanedgelength* **then**
    We look for the closestpoint in the neighborhood of the 5 closest vertices
    Return closestpointfound
else
    We perform an exact search on a hash table of triangle faces
    Return closestpointfound

3.4 Symmetric Matching for Better Accuracy

3.4.1 Introduction

Having improved the complexity of the algorithm we naturally want to improve the quality of the correspondences, since it will be a determining factor of our face recognition system. The accuracy of the correspondences computed with the algorithm described in section 3.2 is too low to capture the finer details of a face.

[2] It is possible to reduce this further by setting the threshold of the edge comparison. Thus we can achieve a trade-off between speed and accuracy.

We observe in figure 3.3(c) that the final approximation $\mathbf{C}(\mathcal{A})$ still bears many similarities with the source mesh, of which it is a deformation (for instance the shape of the nose is closer to $\mathcal{A}$ than $\mathcal{B}$). If we try to diminish the structure term (i.e., using a much smaller α) in equation 3.2 to allow a closer approximation (hereby reinforcing the similarity term) then the result becomes increasingly spurious and the final approximation fails. This is because there is nothing to guide the minimization to the other mesh and help $\mathcal{A}'$ acquire its characteristics.

Need for symmetric correspondences. Another shortcoming of the algorithm 3.2.1 is that the operation of computing the correspondences is not symmetric. This was mentioned by Shelton in [26], without giving a solution. Matching $\mathcal{A}$ to $\mathcal{B}$ or $\mathcal{B}$ to $\mathcal{A}$ will produce two different correspondence fields. This problem is in fact linked with the above: a perfect algorithm should produce an approximation $\mathbf{C}(\mathcal{A})$ so close to $\mathcal{B}$ that by applying its inverse on $\mathcal{B}$, we would obtain again the source mesh $\mathcal{A}$, i.e. $\mathbf{C}^{-1}(\mathcal{B}) \simeq \mathcal{A}$. Therefore, intuitively, if we find a solution to make the computation of the correspondences more symmetric we will also obtain a closer approximations of the target, and thus improve the overall correspondences.

3.4.2 A Symmetric Matching Scheme

Here we describe a symmetric scheme for finding the correspondences. The key idea is that if we want get a closer approximate of the target, then we should deform the target to match the source as well. While retaining its structure, the target mesh will gradually come closer to the source, making it easier to find the correspondences. Updating the two meshes simultaneously will yield two corre-

spondence fields, which have to be combined to obtain the desired correspondence function (from the source mesh to the target mesh). Hence our scheme can be divided in two phases: a symmetric matching phase, and a phase where we combine the two correspondence fields into one.

Phase 1: The symmetric matching. In practice this can be achieved by alternatively matching $\mathcal{A}$ to $\mathcal{B}$, then $\mathcal{B}$ to $\mathcal{A}$, updating the mesh after each iteration and decreasing the strength of the structure term (as discussed in section 3.2). For this purpose the same equation 3.2 can be re-used for the matching process if we rewrite it as follows:

$$E_{\mathcal{A}\to\mathcal{B}}(\mathbf{C}_1) = E^{\mathcal{AB}}_{similarity}(\mathbf{C}_1) + \alpha \cdot E^{\mathcal{A}}_{structure}(\mathbf{C}_1) + \beta \cdot E^{\mathcal{A}}_{smoothness}(\mathbf{C}_1) \qquad (3.9)$$

The new indices and exponents emphasize which mesh is the target or the source in a matching phase. In equation 3.9, $\mathcal{A}$ is the source and $\mathcal{B}$ the target, therefore $E^{\mathcal{A}}_{structure}(\mathbf{C})$ is the structure term of $\mathcal{A}$ and $E^{\mathcal{A}}_{smoothness}(\mathbf{C}_1)$ the smoothness regularizing term for $\mathcal{A}$. Similarly we have,

$$E_{\mathcal{B}\to\mathcal{A}}(C_2) = E^{\mathcal{AB}}_{similarity}(\mathbf{C}_2) + \alpha' \cdot E^{\mathcal{B}}_{structure}(\mathbf{C}_2) + \beta' \cdot E^{\mathcal{B}}_{smoothness}(\mathbf{C}_2) \qquad (3.10)$$

Hence we would match $\mathcal{A}$ to $\mathcal{B}$ minimizing $E_{\mathcal{A}\to\mathcal{B}}(\mathbf{C}_1)$, then $\mathcal{B}$ to $\mathcal{A}$ by minimizing $E_{\mathcal{B}\to\mathcal{A}}(C_2)$. Note that $\mathbf{C}_1$ is defined over $\mathcal{A}$ while $\mathbf{C}_2$ is defined over $\mathcal{B}$. However this cannot be indefinitely repeated as the process would converge to an arbitrary intermediate mesh when the structure terms α and α' become negligible. Instead we stop it after a few iterations when the meshes are much closer without much distortion (i.e., we stop when the structure terms are below a threshold).

We define $\mathcal{A}'$ and $\mathcal{B}'$ as the two updated meshes at this stage, and $\mathbf{C}_{\mathcal{B}'}$ and $\mathbf{C}_{\mathcal{A}'}$ as the correspondence functions.

Phase 2: Obtaining the correspondences. We have $\mathcal{A}' = \mathbf{C}_{\mathcal{B}'}(\mathcal{A})$ and $\mathcal{B}' = \mathbf{C}_{\mathcal{A}'}(\mathcal{B})$, by definition and want to obtain the correspondences $\mathbf{C}_{\mathcal{A}\to\mathcal{B}}$ from $\mathcal{A}$ to $\mathcal{B}$ using $\mathbf{C}_{\mathcal{B}'}$ and $\mathbf{C}_{\mathcal{A}'}$. If $\mathcal{A}'$ and $\mathcal{B}'$ were identical, i.e., if we had obtained a perfect convergence, we would like to write[3] $\mathbf{C}_{\mathcal{A}\to\mathcal{B}} = \mathbf{C}^{-1}_{\mathcal{A}'}(\mathbf{C}_{\mathcal{B}})$ but, as mentioned above the convergence will not be perfect. Nevertheless $\mathcal{A}'$ and $\mathcal{B}'$ should be very close to each other, so that we can attempt to *project* $\mathcal{A}'$ on $\mathcal{B}'$ by minimizing 3.9 with $\alpha = 0$. We could **not** do this earlier because the two meshes were too different, but $\mathcal{A}'$ and $\mathcal{B}'$ are much closer *to one another* as a result of the symmetric matching stage. After the minimization, we obtain a better approximation $\mathcal{A}'' = \mathbf{C}''_{\mathcal{B}}$ and we can use it to compute $\mathbf{C}_{\mathcal{A}\to\mathcal{B}} = \mathbf{C}^{-1}_{\mathcal{A}''}(\mathbf{C}''_{\mathcal{B}'})$. Finally, the best approximation of $\mathcal{B}$, $\mathcal{A}^*$, is obtained with:

$$\mathcal{A}^* = \mathbf{C}^{-1}_{\mathcal{A}''}(\mathbf{C}''_{\mathcal{B}'}(\mathcal{A})) \tag{3.11}$$

We summarize the symmetric matching algorithm in 3.4.1[4].

[3]Note that for the implementation we have to be careful since correspondence function are defined over a mesh and return arbitrary points in space, hence their definition has to be extended so that their inverse can be applied to another mesh with a different number of vertices. But this can be easily done with Hoppe's barycentric coordinates [14].

[4]For more clarity we did not reproduce the inner loop for the minimization of E from algorithm 3.2.1, involving the computation of a closestpoint etc. Instead we use the keyword `Minimize` to symbolize it.

Algorithm 3.4.1: Computation of Correspondences with Symmetric Matching.

Data: Source Mesh $\mathcal{A}$ and target mesh $\mathcal{B}$
Result: Correspondences function $\mathbf{C}$

Initialize the structure coeff.: $\alpha \leftarrow \alpha_0$
Initialize the current approx. meshes: $A' \leftarrow A$ and $B' \leftarrow B$

PHASE I **repeat**
  `Minimize`$(E_{\mathcal{A}' \to \mathcal{B}'}(\mathbf{C}_1), \alpha)$ /* using equation 3.9 */
  `Update`$(\mathcal{A}')$
  `Minimize`$(E_{\mathcal{B}' \to \mathcal{A}'}(\mathbf{C}_2), \alpha)$ /* using equation 3.10 */
  `Update`$(\mathcal{B}')$
  `DecreaseStructureCoeff`(α) /* $\alpha \leftarrow \alpha * AnnealingRate$ */
until α *is below a threshold*

PHASE II `Minimize`$(E_{\mathcal{A}' \to \mathcal{B}'}(\mathbf{C}_1), 0)$ /* here $\alpha = 0$ */
`Update`$(\mathcal{A}')$
$\mathcal{A}^* \longleftarrow \mathbf{C}^{-1}_{\mathcal{A}'}(\mathbf{C}''_{\mathcal{B}'}(\mathcal{A}))$ /* using equation 3.11 */
$\mathbf{C} \longleftarrow$ `GetDisplacements`$(\mathcal{A}, \mathcal{A}^*)$

3.4.3 Experimental Results

To assess the validity of our approach, we compare the correspondence with and without using our symmetric matching scheme. There is no exact measure to assess the validity of a correspondence field since the solution is not unique. However a direct comparison of the final approximations obtained with Shelton's algorithm and our modified symmetric matching scheme shows that our results are undeniably more accurate (see figure 3.3). Our algorithm was tested on several pairs of mesh and the results are better than those obtained using Shelton's algorithm presented in section 3.2. Most notably, the face appears less "blurred", the edges are sharper and some finer features that the previous algorithm could not capture are present with our technique (like the nostrils). However some features are not well approximated like the left ear in figure 3.3.

We also included a comparison of the wire frame models of the approximations

in figure 3.4. The wire frame representation clearly shows that our algorithm produces smoother results.

In the remainder of this thesis we use the symmetric matching algorithm for computing correspondences, the first step of creating a morphable model. In section 5.1, we present a quantitative comparison of the results Shelton's algorithm for the task of impostor detection.

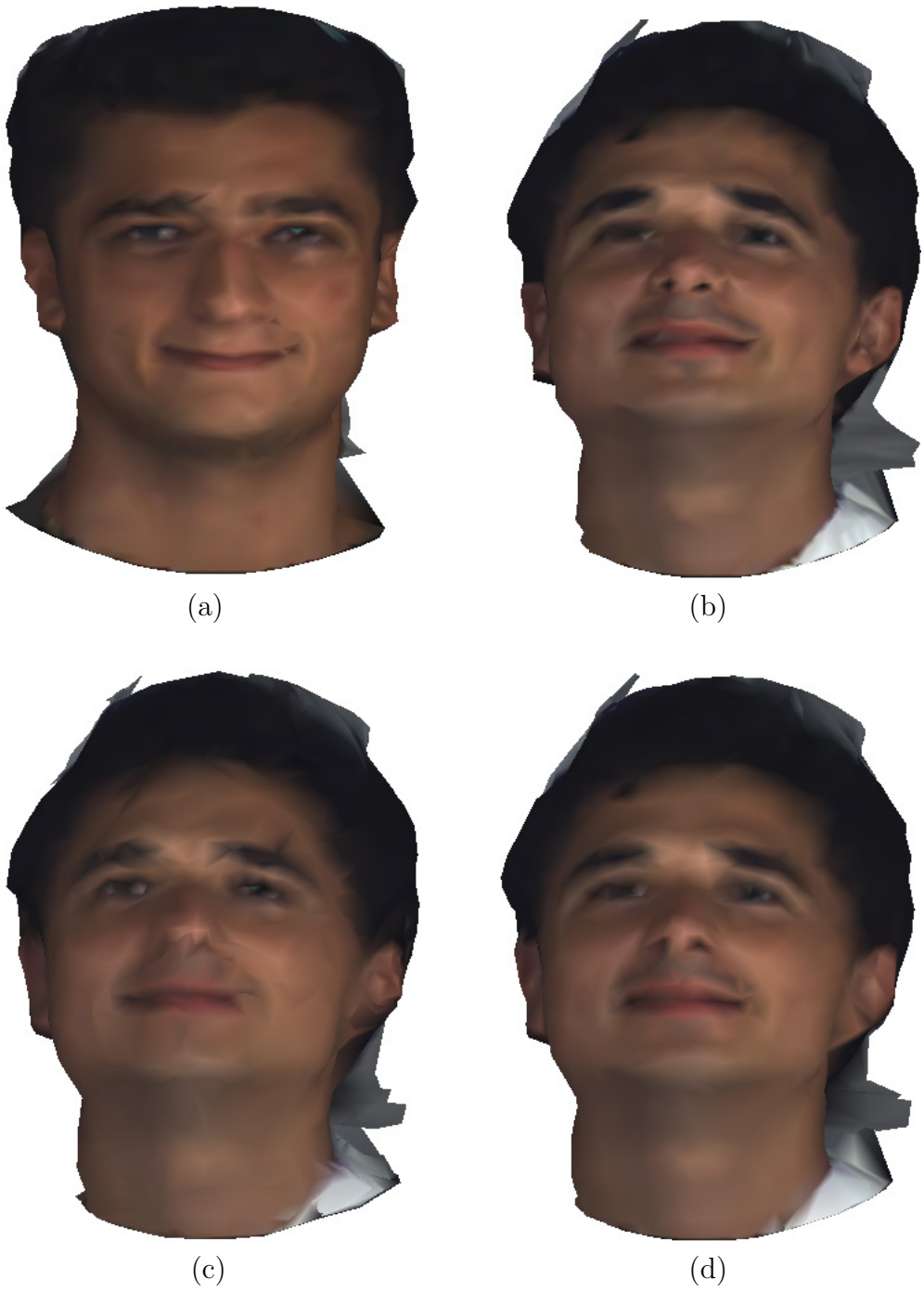
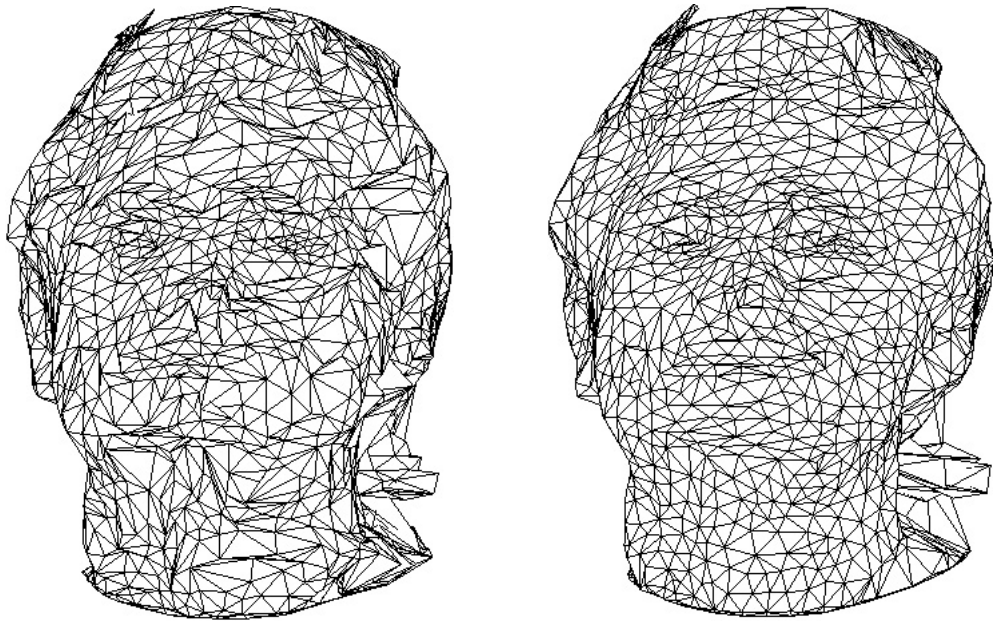


Figure 3.3: Improving the correspondences with symmetric matching. Figures (a) and (b) are the source and target mesh respectively. Figures (c) and (d) are obtained together with the correspondence fields by gradually deforming the source mesh (a) to match (b). (c) was obtained using Shelton's original algorithm [27] and (d) with our symmetric matching algorithm.



(a) (b)

Figure 3.4: Comparison of the wireframes models of the approximations of figure 3.3.Figures (a) and (b) were obtained computed with Shelton's original algorithm and our symmetric matching algorithm respectively. Our results are much smoother, although we use the same number of vertices(1236).

Chapter 4

Construction of Integrated Morphable Models

We can characterize a surface mesh representing a human face according to two attributes that define its appearance: its *identity*, and the (facial)*expression*. The identity concerns the entire structure of the mesh and is unique for each person, whereas the expression modifies finer details of the face (smiles, frowns, etc.). Those two properties are intrinsic to any arbitrary 3D face at a given time. In contrast, the pose and illumination modifying a face are extrinsic since they depend on the environment. In this work, we need to classify human faces according to their identity[1], yet we still need to account for variations due to the other attribute, i.e., facial expressions. This is achieved by our proposed *Integrated morphable models*, a refinement of the morphable models.

In the next section, we describe a basic morphable model and how to deform it linearly to match arbitrary input faces by morphing. In section 4.2, we introduce the IMM, which is used for classification chapter 5. In section 4.3, a filtering technique to further enhance the robustness of the IMM is described.

[1]Nevertheless, expression recognition would be a natural extension of our study. This is discussed in the "Future Works" section of chapter 7

4.1 A Basic Morphable Model

4.1.1 Definitions

A morphable model consists of a *base mesh* and a set of displacement vectors that can be added to this base model to "morph" it into a new surface mesh. There are at least two different representations of morphable models. Vetter and Poggio [32] use two correspondence vectors for shape and texture, whereas Shelton [27] uses a unique 6D vector for both attributes. We follow Shelton's representation which is refined in the next chapter.

In the previous chapter we showed how to compute the correspondence field $\mathbf{C}$ between two meshes $\mathcal{A}$ and $\mathcal{B}$. A basic *morphable model* consists of $\{\mathcal{A}, \mathbf{C}\}$, where $\mathbf{C}$ is simply a set of $N_{\mathcal{A}}$ (number of vertices of mesh $\mathcal{A}$) 6D vectors. As before, $\mathbf{C}(\mathcal{A})$ is the approximation to $\mathcal{B}$ found during the matching process, so we could also represent the morphable model as $\{\mathcal{A}, \mathbf{C}(\mathcal{A})\}$. The representations $\{\mathcal{A}, \mathbf{C}\}$ and $\{\mathcal{A}, \mathbf{C}(\mathcal{A})\}$ will be used interchangeably in this thesis.

Morphing the model. A new surface mesh can be generated by linear combinations of meshes in a morphable model. In our basic example $\{\mathcal{A}, \mathbf{C}\}$ with two meshes this is equivalent to:

$$\mathcal{M} = \mathcal{A} + \alpha_0 \mathbf{C}(\mathcal{A}) \tag{4.1}$$

where α_0 can be any real number. However one can expect that when $\alpha \gg 1$ the mesh will be too deformed to be a recognizable human face.

We can choose any mesh for base mesh, but instead of the first mesh we may want to use an "averaged" face ($\mathcal{O} = \frac{1}{2}\mathcal{A} + \frac{1}{2}\mathbf{C}(\mathcal{A})$), which leads to:

$$\mathcal{M} = \mathcal{O} + \frac{1}{2}\mathcal{A} + (\alpha_0 - \frac{1}{2})\mathbf{C}(\mathcal{A}) \tag{4.2}$$

$$\Longleftrightarrow \mathcal{M} = \mathcal{O} + \alpha_0'\mathbf{C}'(\mathcal{A}) + \alpha_1'\mathbf{C}(\mathcal{A}) \tag{4.3}$$

where for convenience we introduce α_1' and $\mathbf{C}'$ (here $\mathbf{C}'$ is the identity mapping) so that all faces in the database are represented the same way.

Generalization. The generalized morphable model can thus be described as:

$$\{\mathcal{O}, \mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_n\} \tag{4.4}$$

where $\mathcal{O}$ is the base surface and $\mathbf{C}_1, \ldots, \mathbf{C}_n$ are the n correspondence fields. A new surface mesh is generated by a linear combination of the examples in the morphable model:

$$\mathcal{M} = \mathcal{O} + \sum_{i=0}^{n} \alpha_i \mathbf{C}_i(\mathcal{O}) \tag{4.5}$$

Hence any mesh generated with the morphable model is uniquely described by a set of $\alpha_0 \ldots \alpha_n$ morphing parameters, and each α_i accounts for an example face in the database i.e., the α_i are weights and equation 4.5 is a weighted average. In vector notation, the *morphing vector* $\vec{\alpha} = (\alpha_0 \ldots \alpha_n)$ uniquely determines the mesh $\mathcal{M}$.

Thus, by morphing the model to fit an arbitrary face, we obtain a set of α_i parameters. These $\alpha_\mathbf{i}$ parameters can be used to recognize the input face. A method for deforming the model to optimally fit an input mesh is described in the next subsection.

4.1.2 Morphing the Model to Fit an Arbitrary Surface

Matching the model to an arbitrary mesh can be performed as in chapter 3, where we minimize an energy functional defined over two meshes to deform one to fit the other and find the correspondences. The difference between this matching and the correspondence computation described in chapter 3 is that an energy function has to minimized over the α parameters and not over the vertices of the mesh. Only the $E_{similarity}$ term from equation 3.2 is needed in the energy function E, together with a regularizing parameter to penalize high α values:

$$E(\vec{\alpha}) = \int_{\mathcal{I}} \|\vec{i} - \vec{P}_{\mathcal{M}_{\vec{\alpha}}}(\vec{i})\|^2 d\vec{i} + \int_{\mathcal{M}} \|\vec{m} - \vec{P}_{\mathcal{I}}(\vec{m})\|^2 d\vec{m} + \lambda \sum_{j=0}^{n} \alpha_j \qquad (4.6)$$

where: $\mathcal{I}$ is the input mesh, $\vec{\alpha}$ is the morphing vector, $\mathcal{M}$ is defined in equation 4.5, and $\lambda \sum_{j=0}^{n} \alpha_j$ is the regularizing term. Once again the terms of this equation are quadratic and we can use standard methods like conjugate gradient for a fast resolution. The detailed formulation of the solution of this least-square problem can be found in [27].

4.2 Building Integrated Morphable Models

The morphable model described in section 4.1 can be used to model a set of 3D faces, match an arbitrary input face, and classify it using the resulting α_i parameters (using the euclidean distance on the α vector for instance). However with only one face per individual in the model, only about 60% of our input faces could be recognized.

Indeed with only one face per individual, the model can only capture the *extrapersonal* variations, i.e., the variations from one individual to another, but is unable to store *intrapersonal* variations, or the variations of each single individual's

face under different expressions. To solve this we can add more faces of the same individuals, preferably with different expressions. In this work, we were able to obtain good results using as little as two 3D faces per individual, as demonstrated in section 5.3.

4.2.1 Definition of an IMM

In section 4.1, the correspondences fields in the morphable model were not ordered in any particular fashion (they were all considered equivalent). In this section we aim to make a distinction between intrapersonal and extrapersonal correspondence fields, thus creating an *integrated morphable model* (IMM).

Our approach, which is holistic because it does not rely on feature extraction, involves first creating morphable models for each individual using at least two example faces per person. These morphable models i.e, *submodels*, are then merged into a global model that is used for recognition. The process is summarized in figure 4.1 and the algorithm is detailed in section 4.2.2.

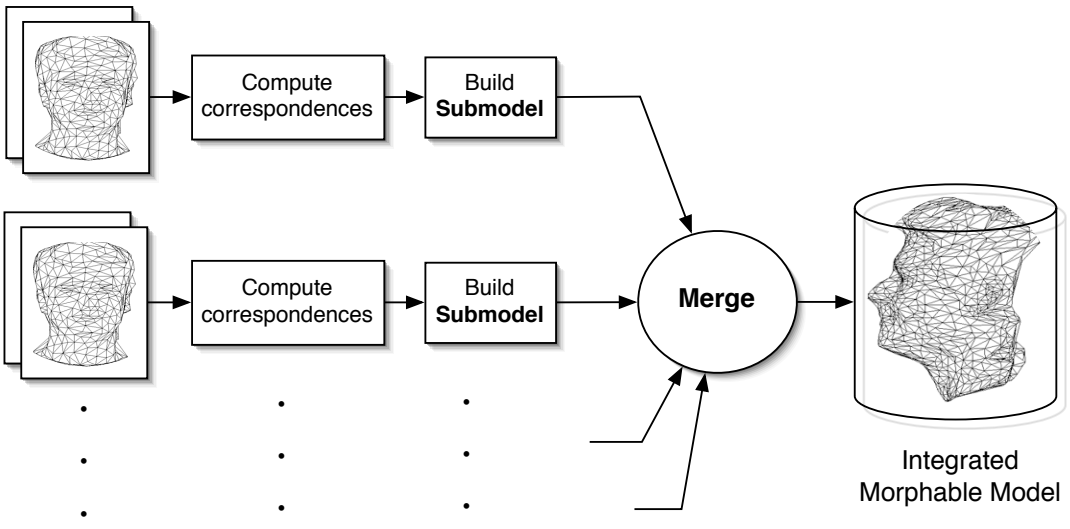


Figure 4.1: Overview of the integrated morphable model creation.

Motivation. The correspondence fields obtained at the submodel level (see figure 4.1) approximate *expression changes* in an individual, whereas the correspondences obtained during the merging represent *identity changes*. Once all meshes are set in correspondence, **we can change the facial expression of any mesh to that another mesh**, which will be useful for our goal of expression-invariant recognition. We refer to this property as *expression synthesis*. This property is demonstrated in section 5.2.1 and verified experimentally in section 5.3. Besides the ability to synthesize expressions, the IMM has other advantages:

- more accurate correspondences than a basic morphable model; it is easier to compute correspondences between two faces of the same person (in the submodels) than between two different individuals, since they are much more similar.

- artifacts created by the 3D capture device can be localized and filtered out using the submodels, as described in 4.3. This filtering enhances the robustness of the morphable model, which would otherwise accumulate errors (see section 4.3).

An IMM is made of a base mesh and sets correspondence fields related to a single individual (inherited from the submodels):

$$\widehat{\mathcal{G}} = \{\mathcal{B}_{\widehat{\mathcal{G}}}, \{\mathbf{C}_1^1, \dots \mathbf{C}_1^{n_1}\}, \cdots, \{\mathbf{C}_M^1, \dots, \mathbf{C}_M^{n_M}\}\} \tag{4.7}$$

where $\mathcal{B}_{\widehat{\mathcal{G}}}$ is the base mesh, C_i^j is the j^{th} correspondence field of submodel i, M is the number of submodels (or number of individuals), and n_i the number of correspondence field in submodel i.

If we use only two meshes per person (as in our experiments in section 5.3), this equation is simplified:

$$\widehat{\mathcal{G}} = \{\mathcal{B}_{\widehat{\mathcal{G}}}, \{\mathbf{C}_1^1, \mathbf{C}_1^2\}, \cdots, \{\mathbf{C}_M^1, \mathbf{C}_M^2\}\} \tag{4.8}$$

A IMM is similar to the morphable models of section 4.1, except for the creation process and the semantics (the meaning of its components). Therefore we can still use the morphing technique of section 4.1.2 to match an input face.

4.2.2 Construction of IMMs

IMMs are constructed in two stages, as illustrated in figure 4.1. In the first stage, we create morphable models i.e., submodels, for each individual in the database (at least two meshes per person). In the second, stage we merge all the submodels in a global morphable model: the IMM.

The merging phase involves computing the correspondence fields between meshes extracted from the submodels. We detail this phase for submodels with two correspondences fields but it is easily extensible to submodels with an arbitrary number of correspondence fields.

We start with two submodels:

$$\widehat{\mathcal{S}}_1 = \{\mathcal{B}_1, \{\mathbf{C}_1(\mathcal{B}_1), \mathbf{C'}_1(\mathcal{B}_1)\}\} \tag{4.9}$$

$$\widehat{\mathcal{S}}_2 = \{\mathcal{B}_2, \{\mathbf{C}_2(\mathcal{B}_2), \mathbf{C'}_2(\mathcal{B}_2)\}\} \tag{4.10}$$

where $\mathcal{B}_1$ and $\mathcal{B}_2$ are the base meshes and $\mathbf{C}_i, \mathbf{C}'_i$ are correspondences fields.

We compute the correspondence field $\mathbf{C}''$ between the base meshes of the two submodels, so that $\mathbf{C}''(\mathcal{B}_1) = \mathcal{B}'_2$ is an approximate mesh of $\mathcal{B}_2$. $\mathbf{C}''$ represents the extrapersonal variations between $\widehat{\mathcal{S}}_1$ and $\widehat{\mathcal{S}}_2$. These correspondence fields are

combined with those of the submodels (representing intrapersonal variations) to yield a IMM:

$$\widehat{\mathcal{S}}_{12} = \{\mathcal{B}_1, \{\mathbf{C}_1(\mathcal{B}_1), \mathbf{C'}_1(\mathcal{B}_1)\}\{\mathbf{C}_2(\mathbf{C''}(\mathcal{B}_1)), \mathbf{C'}_2(\mathbf{C''}(\mathcal{B}_1))\}\} \tag{4.11}$$

We continue this process by similarly merging the next submodel $\widehat{\mathcal{S}}_3$ to $\widehat{\mathcal{S}}_{12}$ yielding $\widehat{\mathcal{S}}_{123}$, etc. Once we merged all submodels, we obtain a IMM $\widehat{\mathcal{S}}_{1\cdots n}$ that contains all the correspondence fields of the submodels.

An important step is the choice of the first submodel $\widehat{\mathcal{S}}_1$. We can choose the submodel with the highest number of vertices for better accuracy or the one with less artifacts (technique for detecting artifacts are discussed in the following section).

The IMM construction algorithm is given below (algorithm 4.2.1). Figure 4.2 illustrates the morphing ability of an IMM constructed with 6 faces of 3 persons (2 faces per person).

Algorithm 4.2.1: Merging submodels into a global integrated model.

Data: Submodels $\widehat{\mathcal{S}}_0 \ldots \widehat{\mathcal{S}}_n$, their respective base meshes $\mathcal{B}_i$.
Result: A global IMM $\widehat{\mathcal{G}}$ and its base mesh $\mathcal{B}_{\widehat{\mathcal{G}}}$.

$\widehat{\mathcal{G}} \longleftarrow \widehat{\mathcal{S}}_0$ /* assuming $\widehat{\mathcal{S}}_0$ has the max. number of vertices. */
foreach *Submodel* $\widehat{\mathcal{S}}_i$ **do**
  $\mathbf{C} \leftarrow$ `FindCorresp`$(\mathcal{B}_{\widehat{\mathcal{G}}}, \mathcal{B}_i)$ /* described in chapter 3. */
  forall *Correspondence field* $\mathbf{C}_i$ *in* $\widehat{\mathcal{S}}_i$ **do**
    Add correspondence $(\mathbf{C}_i(\mathbf{C}))$ **to** model $\widehat{\mathcal{G}}$
  Update $\mathcal{B}_{\widehat{\mathcal{G}}}$ to account for the new correspondences in $\widehat{\mathcal{G}}$
return $\widehat{\mathcal{G}}$

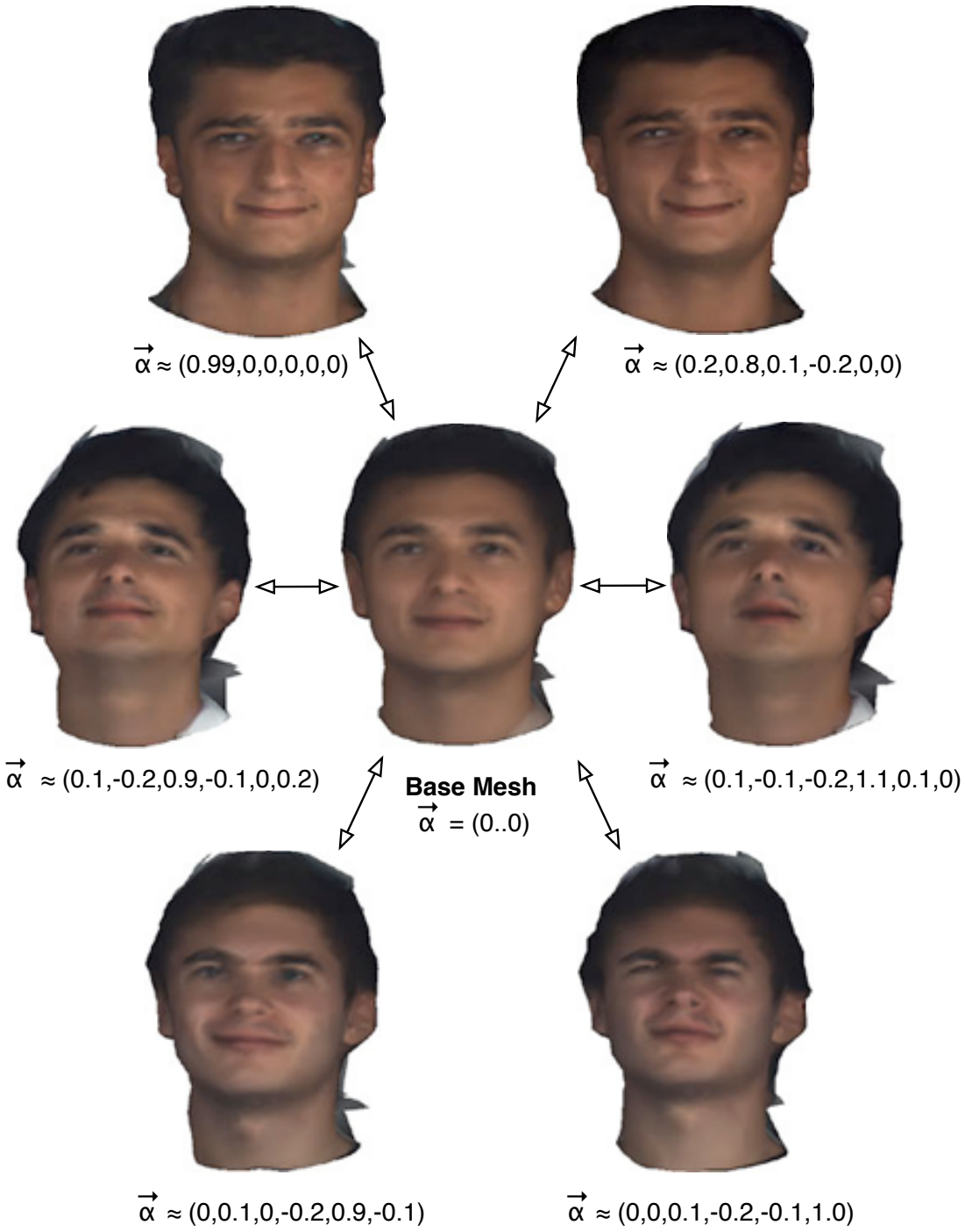


Figure 4.2: **Morphing a model.** We created a integrated morphable model (see section 4.2) with 6 example faces of 3 persons. When modifying the morphing parameters α_i in equation 4.5, the model morphs the base mesh into different faces. When $\vec{\alpha}$ is of the form $(0 \cdots 0, 1, 0 \cdots 0)$ we obtain an approximation of one of the example faces. If α is arbitrary then the model becomes a weighted average of these meshes, defined by equation 4.5.

4.3 Filtering out 3D Reconstruction Errors

The 3D reconstruction performed by modern stereo capture device is a noise-sensitive process that relies heavily on the quality of images output by the cameras. Even small amounts noise in the original image can produce major artifacts in the reconstruction. There are mostly two types of artifacts in 3D meshes: "holes" and "spikes" [7]. Holes were not observed in all our datasets but spikes were observed (see figure 4.3) so we focus on eliminating spikes .

Artifacts can lead to inaccurate correspondence fields. Algorithm 3.2.1 attempts to find points corresponding to the artifacts although they are not relevant features. The resulting IMM will contain errors and become increasingly inaccurate as more submodels are merged (see section 4.2.2). This in turn leads to errors when the model is morphed to fit an input face. Therefore, artifacts have to removed to make the model more robust and to prevent the accumulation of errors.

In this section, we describe a method to remove the artifacts at the submodel level which is not applicable to a basic morphable model but only to IMMs. It involves first comparing a meshes extracted from the submodel to locate the potential artifacts and then "filtering" the relevant regions.

4.3.1 Locating Artifacts

The original morphable model framework [30] offers no solution to correct these reconstruction errors. The correspondence computation is a "blind" process and all parts of the face are matched, whether they are relevant features or artifacts. If we try to locate the artifacts by selecting the points with highest curvature then

we may end up removing important features of the face like the nose, ears etc. which also have high curvature.

However artifacts can be located more precisely by comparing two meshes of the same person. Since spikes are essentially random they will not be identical across different captured meshes. The meshes can only be compared vertex by vertex if they were set in correspondence, which is the case for a submodel.

Hence, given a submodel $\widehat{\mathcal{S}} = \{\mathcal{O}, \{\mathbf{C}(\mathcal{O}), \mathbf{C}'(\mathcal{O})\}\}$, we compare its meshes $\mathbf{C}(\mathcal{O}) = \mathcal{A}$ and $\mathbf{C}'(\mathcal{O}) = \mathcal{B}$. For each triangle face of $\mathcal{A}$ we compare its orientation with the orientation of corresponding triangle face of $\mathcal{B}$, (by evaluating the scalar product of their normals). Intuitively, the local orientation will also change due to differences in facial expressions. But this orientation change will be very high for a spike, which is completely random.

We select triangles of the periphery of the face that have the highest orientation change. This is consistent with the observation that artifacts generally appear near the boundary of the surface, where the 3D reconstruction is most difficult. We obtain a set of scattered points on the surface.

4.3.2 Surface Segmentation

Designing an accurate segmentation scheme that separates the artifacts from the rest of the face would be an extremely difficult task. We have implemented algorithms which use the curvature properties of the mesh like "watershed segmentation" [17], but they fail to give satisfactory results. This is primarily because these algorithms were originally designed for CAD models and are not suited to human faces. Indeed 3D segmentation is still an active area of research. The set of points obtained in subsection 4.3.1 have a high probability of being part of a

spike but this set does not include all of the spikes: because of their random nature some of them may have the same orientation across different example faces and they would not be detected in the previous phase.

Instead we opted for a simpler scheme where we do not attempt exact segmentation. We do not look for surface patches that coincides with the artifacts, but rather regions large enough to *contain* the artifacts. Thus, the problem becomes much easier and typical region growing techniques can be used to obtain a few surface patches from the set of scattered points. We discard the regions that have very few vertices and obtain a set of regions that contain all possible artifacts.

4.3.3 Selectively Removing the Artifacts

Once we obtain regions containing potential artifacts, they cannot be simply suppressed (by removing the vertices) as it would create holes. Flattening the regions might also damage the mesh since they do not coincide exactly with the artifacts, they contain the artifacts as well as regular portions of the mesh. Instead we have to somehow smoothen the spikes while minimizing surface shrinking (a typical drawback of curvature filters). We use two filters described by Meyer et al. in [18]. These filters, *mean curvature operator* and the *umbrella operator*, are based on two operators that can be iteratively applied to a portion of the mesh. The first filter preserve more of the initial geometry of the mesh (and does not distort the mesh) and the second one suppresses the spikes more effectively (but incurs some shrinking of the mesh). Combining the two filters gives better results [18] as figure 4.3 which shows results of the application of our technique on a mesh with a large initial artifact.

Clearly, the filtering results in very little distortion of the original mesh, while

the spike has been effectively leveled. A drawback of this method is that the coefficients (one for each filter) are set by trial and error but they are not changed thereafter. An improvement to the current algorithm would be to adapt the filters coefficient to each mesh instead of using the same values for all meshes.

(a) (b)

Figure 4.3: Filtering artifacts from the capture device. Figure 4.3(a) is the original face, and figure 4.3(b) the same face after filtering with the method described in this section.

Chapter 5

Face Recognition with IMMs

Once an IMM is created, it can be used to recognize an arbitrary input face. The previous chapter describes a method to morph the IMM to the input face which yields an approximation mesh to the input face and a set of morphing parameters ($\vec{\alpha}$). This information is then used to ascertain whether:

a) this input face belongs to one of the n individuals in the IMM: this is the impostor detection phase,

b) the identity of the individual assuming a) was true: this is the identification phase.

Figure 5.1 provides an overview of the whole recognition scheme based on the IMM. The impostor detection and identification phases are discussed in sections 5.1 and 5.2. Finally, the experimental results used to test the validity of our approach are presented in section 5.3 .

5.1 Impostor Detection

Intuitively, an input mesh is recognized as belonging to one of the faces used to create the IMM if its approximation obtained by morphing the IMM (see section 4.1.2) is "close" to the input mesh. If a measure of the distance between the input and approximation meshes is below some threshold, then the input mesh is deemed to be close enough to be the face of a person in the model.

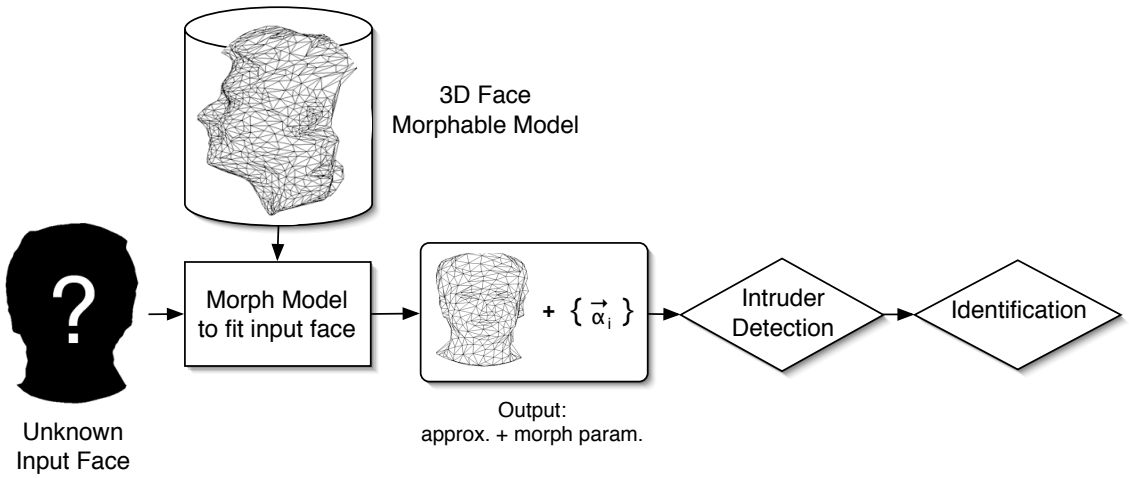


Figure 5.1: 3D Face recognition stage

The meshes of imposters (i.e., faces that are not used to create the IMM) will result in high distance values because the IMM cannot approximate them accurately. On the other hand, surfaces (i.e., meshes) that represent someone in the IMM will be easily approximated and the distance to their respective approximation should be lower than for an impostor.

Distance computation. The distance between two meshes $\mathcal{A}$ (input) and $\mathcal{B}$ (approximation of $\mathcal{A}$) is computed by measuring the distortion of each point of the mesh when we "backproject" the input face to the approximation mesh and vice-versa. This is similar to the "backward warping" used in [31] to check the closeness of a match between two faces. This procedure is as follows.

Each point $\vec{p}$ of one mesh is projected onto the other mesh by finding the closest point (as in chapter 3), and then we project it back to the original mesh, which yields $\vec{q} = \vec{P}_{\mathcal{A}}(\vec{P}_{\mathcal{B}}(\vec{p}))$. If the approximation was perfect, the backprojected point should return to its original location i.e., $\vec{p} = \vec{q}$, but since it is not usually the case it will be at some distance from its original location. The distance $\|\vec{p} - \vec{q}\|$ for all

points of the mesh is computed, and these distance values are averaged to give a measure of the "closeness" of the approximation. This is done for both meshes (i.e, $\mathcal{A}$ to $\mathcal{B}$ then $\mathcal{B}$ to $\mathcal{A}$) to provide a fair estimate. We summarize this operation in the following equation (with the usual notation):

$$d_{\mathcal{A},\mathcal{B}} = \frac{1}{N_{\mathcal{A}}} \sum_{i=1}^{N_{\mathcal{A}}} \|\vec{p_i} - \vec{P}_{\mathcal{A}}(\vec{P}_{\mathcal{B}}(\vec{p_i}))\| + \frac{1}{N_{\mathcal{B}}} \sum_{j=1}^{N_{\mathcal{B}}} \|\vec{p_j'} - \vec{P}_{\mathcal{B}}(\vec{P}_{\mathcal{A}}(\vec{p_j'}))\| \qquad (5.1)$$

where $\vec{p_i}$ and $\vec{p_j'}$ are sampled points on the meshes $\mathcal{A}$ and $\mathcal{B}$, respectively.

Classification. Once this distance d is computed between an input mesh and its morphed approximation, it is compared to a threshold to decide if the input belongs one of the persons used to create the IMM. In our work, the threshold is set by trial and error. If some of the morph parameters are too high (for instance > 2), the morphed mesh (i.e., approximation mesh) will probably be distorted which indicates that the matching has failed (as explained in Chapter 4). In the following section, we present a method to identify faces and give a confidence margin which indicates the quality of the fit. Impostors typically have a low confidence margin and we can use it to detect them using a threshold. The impostor detection method (algorithm 5.1.1) is summarized below.

The quality of the approximation obtained by morphing the IMM (see section 4.1.2) and therefore the quality of our correspondence algorithm is crucial. If the computed correspondence fields used to create the IMM are too inaccurate, morphing the IMM will not yield a good approximation to an input mesh and the distance values computed by algorithm 5.1.1 will be high even for non-impostors. It will be difficult to find an appropriate threshold for this distance. Thus, the impostor detection is a good test of the accuracy of the correspondences computed with our symmetric matching algorithm introduced in section 3.4.2. The experimental

Algorithm 5.1.1: Impostor detection.

Data: Input mesh $\mathcal{I}$, and a global model $\widehat{\mathcal{G}}$.
Result: *Decision*: if the mesh belongs to the model

$\{\mathcal{M}_{\widehat{\mathcal{G}}},\ \vec{\alpha}\} \leftarrow$ `MorphModel`$(\widehat{\mathcal{G}}, \mathcal{I})$ /* described in section 4.1.2 */

foreach α_i *in* $\widehat{\mathcal{S}}_i$ **do**
  if $|\alpha_i| > 2$ **then**
    return $\mathcal{I} \notin \widehat{\mathcal{G}}$

if *ComputeDistance*$(\mathcal{I}, \mathcal{M}_{\widehat{\mathcal{G}}}) > threshold$ **then**
  return $\mathcal{I} \notin \widehat{\mathcal{G}}$

$margin \leftarrow$ `ComputeMargin`$(\vec{\alpha})$ /* described in section 5.2.2 */
if *margin is too low* **then**
  return $\mathcal{I} \notin \widehat{\mathcal{G}}$
else
  return $\mathcal{I} \in \widehat{\mathcal{G}}$

results are presented in section 5.3.

5.2 Identification Phase

If an input face is found to be one of the faces integrated in the IMM, the next step would be to identitfy the input face using the vector $\vec{\alpha} = (\alpha_1, \cdots, \alpha_n)$ produced by the morphing operation (see figure 5.1). A classical approach for classifying feature vectors (like $\vec{\alpha}$) would be to compare distances between this vector and those obtained for the meshes integrated in the IMM, to ascertain the minimum distance. A confidence margin could be obtained by evaluating the relative difference between the first and second minimum distances. However this method does not achieve high recognition results, as discussed in section 5.3. We propose a classifier that relies specifically on IMMs by transforming $\vec{\alpha}$ into a new vector $\vec{\alpha}'$ prior to classification (see section 5.2.2). In section 5.3, we compare the performance of our proposed classifier with the minimum-distance classifier.

5.2.1 Interpretation of the Morphing Parameters

In this section, we assume that the IMM was created using two meshes per person i.e., each submodel has two correspondence fields, but it can be easily extended to an IMM with more meshes per person. Let $\widehat{\mathcal{G}}$ be an IMM with M submodels. We have:

$$\widehat{\mathcal{G}} = \{\mathcal{O}, \{\mathbf{C}_1^1, \mathbf{C}_1^2\}, \cdots, \{\mathbf{C}_M^1, \mathbf{C}_M^2\}\} \tag{5.2}$$

where $\mathcal{O}$ is the base mesh, C_i^j is the j^{th} correspondence field of submodel i, and M is the number of submodels (or number of individuals).

Each correspondence field C_i^j in equation 5.2 has a corresponding morphing parameter $\alpha_{j,i}$ (see section 4.1). $\alpha_{j,i}$ collectively form a $2 \times M$ matrix:

$$\hat{\alpha} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,M} \\ \\ \alpha_{2,1} & \cdots & \alpha_{2,M} \end{pmatrix} \tag{5.3}$$

With this notation, the mesh $\mathcal{M}$ generated by the IMM is given by:

$$\mathcal{M} = \mathcal{O} + \sum_{i=1}^{M} (\alpha_{1,i} \cdot \mathbf{C}_i^1(\mathcal{O}) + \alpha_{2,i} \cdot \mathbf{C}_i^2(\mathcal{O})) \tag{5.4}$$

Each term $(\alpha_{1,i} \cdot \mathbf{C}_i^1(\mathcal{O}) + \alpha_{2,i} \cdot \mathbf{C}_i^2(\mathcal{O}))$ in equation 5.4 is a *contribution* to the formation of $\mathcal{M}$. As explained in chapter 4, two attributes uniquely define $\mathcal{M}$; its identity, and its facial expression. We would like to identify the pairs $(\alpha_{1,i}, \alpha_{2,i})$ in equation 5.4 that relate to the identity of mesh $\mathcal{M}$ and to its facial expression. This achieved by the following proposition:

Proposition. Given a pair of morphing parameters $(\alpha_{1,i}, \alpha_{2,i})$ related to submodel i obtained by morphing the IMM into a mesh $\mathcal{M}$, we can categorize the contribution of submodel i to the formation of $\mathcal{M}$ into two types :

- Pairs of the form $(k, -k')$ or $(-k, k')$ which yield a contribution to the *expression* of $\mathcal{M}$.

- Pairs of the form (k, k'), (k, ϵ) or (ϵ, k') which yield a contribution to the *identity* of $\mathcal{M}$.

where $\epsilon \approx 0$ is negligible, $k \approx k' > 0$ and $k \gg \epsilon$.

Indeed, pairs of the first type correspond to the term: $\pm k(\mathbf{C}_i^1(\mathcal{O}) - \mathbf{C}_i^2(\mathcal{O}))$ in equation 5.4 and the two correspondence fields of the same submodel i cancel out. Since $\mathbf{C}_i^1$ and $\mathbf{C}_i^2(\mathcal{O})$ relate to the same person but with different expressions, their difference will contribute to the morph $\mathcal{M}$ with the expression change from mesh $\mathbf{C}_i^1(\mathcal{O})$ to mesh $\mathbf{C}_i^2(\mathcal{O})$.

On the other hand, in the second type, the coefficients do not cancel out; $k\mathbf{C}_{2i-1}(\mathcal{O})$, $k\mathbf{C}_{2i}(\mathcal{O})$ or $k(\mathbf{C}_{2i-1}(\mathcal{O}) + \mathbf{C}_{2i}(\mathcal{O}))$ yield a *positive* contribution. Hence the whole structure of the two meshes $\mathbf{C}_{2i-1}(\mathcal{O})$ and $\mathbf{C}_{2i}(\mathcal{O})$ is added, this term contributes the *identity* of $\mathcal{M}$.

The above proposition supports the claim of *expression synthesis* made in section 4.2.1 that the IMM can change the facial expression of a given person i to that another person j in the model. A new expression (from submodel j) for person i can be synthesized from the IMM by setting:

$$\alpha_{1,i} = \alpha_{2,i} = \frac{k}{2} \quad \textit{and} \quad \alpha_{1,j} = -\alpha_{2,j} = \frac{k'}{2} \tag{5.5}$$

with $k, k' \in [-1, 1]$ and $j \neq i$, in equation 4.5. If all remaining α parameters are negligible then the resulting mesh $\mathcal{M}$ in equation 5.4 will exhibit submodel i's identity (person i) with a facial expression from submodel j.

This is illustrated in figure 5.2 where we created meshes with new expressions

by modifying the alpha parameters according to equation 5.5. The parameters are given in table 5.1.

Figure	Morph Parameters			
5.2(d)	(1.0, 0.0)	(1.0, -1.0)	(0.0, 0.0)	(0.0, 0.0)
5.2(e)	(0.9, 0.0)	(-0.9, 0.9)	(0.0, 0.0)	(0.0, 0.0)
5.2(f)	(1.0, 0.0)	(0.0, 0.0)	(-0.6, 0.6)	(0.0, 0.0)
5.2(g)	(1.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)	(-0.6, 0.5)

Table 5.1: Morphing parameters $\vec{\alpha}$ corresponding to the expression synthesis of figure 5.2

Of course this is only an approximation and the quality of the correspondences are crucial, since most expressions involve a high level of details.

Consequences for classification: In short, this proposition means that the model can **handle the *identity* and *expression* properties of a face independently** and match any subject of the model with any expression as long as there exists an example of this expression in the model.

Naturally, in real-life examples the two categories of pair of morphing parameters described in the proposition are not so clear-cut, mostly because of noise and inaccuracies in the correspondence. Nevertheless it suggests the use of a vector that contains only the identity information. Thus we can compute the average of pairs (α_i^1, α_i^2) so that the terms that account for an expression change will cancel off and only the terms relevant for the identity of the mesh will remain. The resulting reduced vector is as follows:

$$\vec{\alpha}' = (\frac{\alpha_1^1 + \alpha_1^2}{2}, \cdots, \frac{\alpha_M^1 + \alpha_M^2}{2}) = (\alpha_1^{'}, \ldots, \alpha_M^{'}) \tag{5.6}$$

In the following we will use the $\vec{\alpha}'$ to classify the faces.

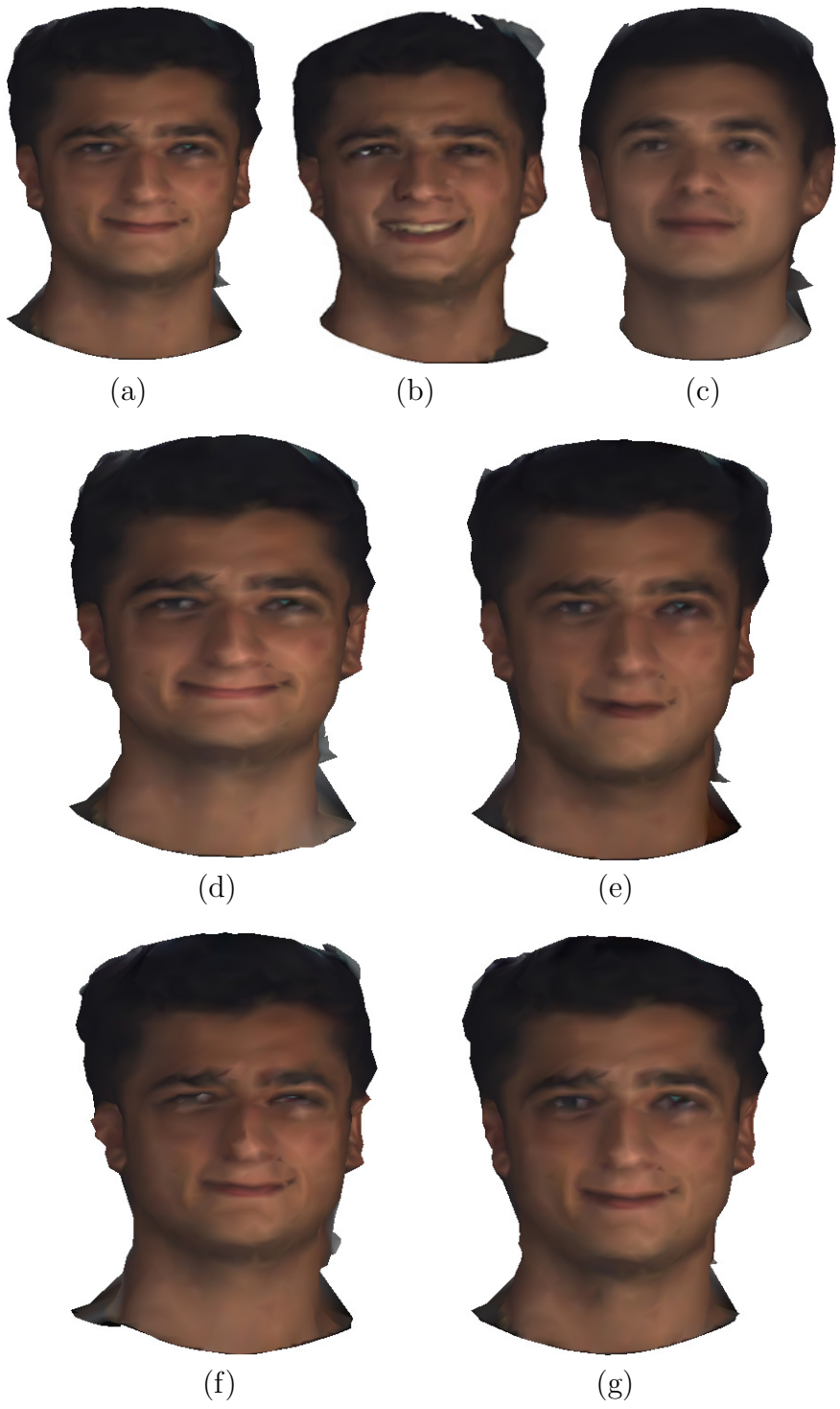


Figure 5.2: The IMM created using meshes (a) and (b) and those of other persons is shown in (c) (its the basis mesh). The following new expressions are synthesized using the IMM: (d) a large contented smile, (e) with his mouth is half opened, (f) frowning, and (g) showing a surprised expression.

5.2.2 Classification Algorithm

In this section, we describe a classification algorithm.

Once it is established that the mesh belongs to one of the persons integrated in the IMM, we can use the reduced vector defined in equation 5.6 to classify the face. The maximum parameter $\alpha_i^{'}$ gives the identity of the mesh. The relative difference with the second maximum $\alpha_j^{'}$ gives us a confidence margin: $\frac{\alpha_i^{'}-\alpha_j^{'}}{\alpha_i^{'}}$. If this value is low then $\alpha_i^{'}$ and $\alpha_j^{'}$ are very close in relative terms, there is no clear maximum and the results are uncertain (i.e, the input mesh might not be recognised). In contrast a high value means that input mesh is very close to submodel i in the IMM. A very low confidence value may also indicate the input mesh was not in the model i.e., it is an impostor (see section 5.1).

The complete algorithm is summarized below (algorithm 5.2.1).

Algorithm 5.2.1: Classification with IMM.

Data: Input mesh $\mathcal{I}$, and a global model $\widehat{\mathcal{G}}$. $\vec{\alpha} = ((\alpha_1, \alpha_2), \cdots, (\alpha_{2n-1}, \alpha_{2n}))$.
Result: *Decision*: if the mesh belongs to the model
if so returns the submodel index and a confidence margin.

$\{\mathcal{M}_{\widehat{\mathcal{G}}},\ \vec{\alpha}\} \leftarrow$ `MorphModel`$(\widehat{\mathcal{G}}, \mathcal{I})$ /* described in section 4.1.2 */

PHASE A
foreach α_i *in* $\widehat{\mathcal{S}}_i$ **do**
  if $|\alpha_i| > 2$ **then**
    return $\mathcal{I} \notin \widehat{\mathcal{G}}$

if *ComputeDistance*$(\mathcal{I}, \mathcal{M}_{\widehat{\mathcal{G}}}) > threshold$ **then**
  return $\mathcal{I} \notin \widehat{\mathcal{G}}$

PHASE B
$\vec{\alpha'} \longleftarrow$ `Reduce`$(\vec{\alpha})$ /* according to equation 5.6 */
$i \leftarrow$ `Max`$(\vec{\alpha'})$ and $j \leftarrow$ `2ndMax`$(\vec{\alpha'})$
return $\mathcal{I} \in Sumodel\ i \mid margin = \frac{\alpha_i^{'}-\alpha_j^{'}}{\alpha_i^{'}}$

5.3 Experimental Results

5.3.1 Introduction

We created a dataset of human faces by getting subjects to express different emotions from a set of 6 basic emotions: happiness, sadness, anger, surprise, disgust or fear and a neutral expression. Those 6 emotions are sometimes considered to be "universal" [11] but is still debated [22].

3D meshes of the faces were captured using a stereo digitizer (a geometrix FaceVision device [1]). Subjects were free to choose which emotion to express but they were asked to change their facial expression so that a diverse dataset can be obtained. Some faces in the database were taken over an interval of several months and the subjects also have different hair styles. The faces used in our experiments are presented in the Appendix.

A total of 82 faces were captured for 19 subjects (1 to 8 meshes per subject). When 3 meshes or more are available for a given subject, two of them can be used to create an IMM (usually one with a neutral expression and any other one from the 6 emotions mentioned above) and the remaining meshes form a test set for recognition. Figure 5.3 shows an example of various faces captured for single subject. Meshes that were not used to create the model could be used as "impostors" (see section 5.1) to test the impostor detection phase.

Although the database consists of frontal views, some subjects may be looking at different directions (up, down...) and no attempt was made to correct or align the meshes. The meshes have around 1200 vertices, which is lower than many other works, but we show in section 5.3.3 that it is sufficient to achieve high recognition rate. Indeed many works rely on the computation of curvature properties (as

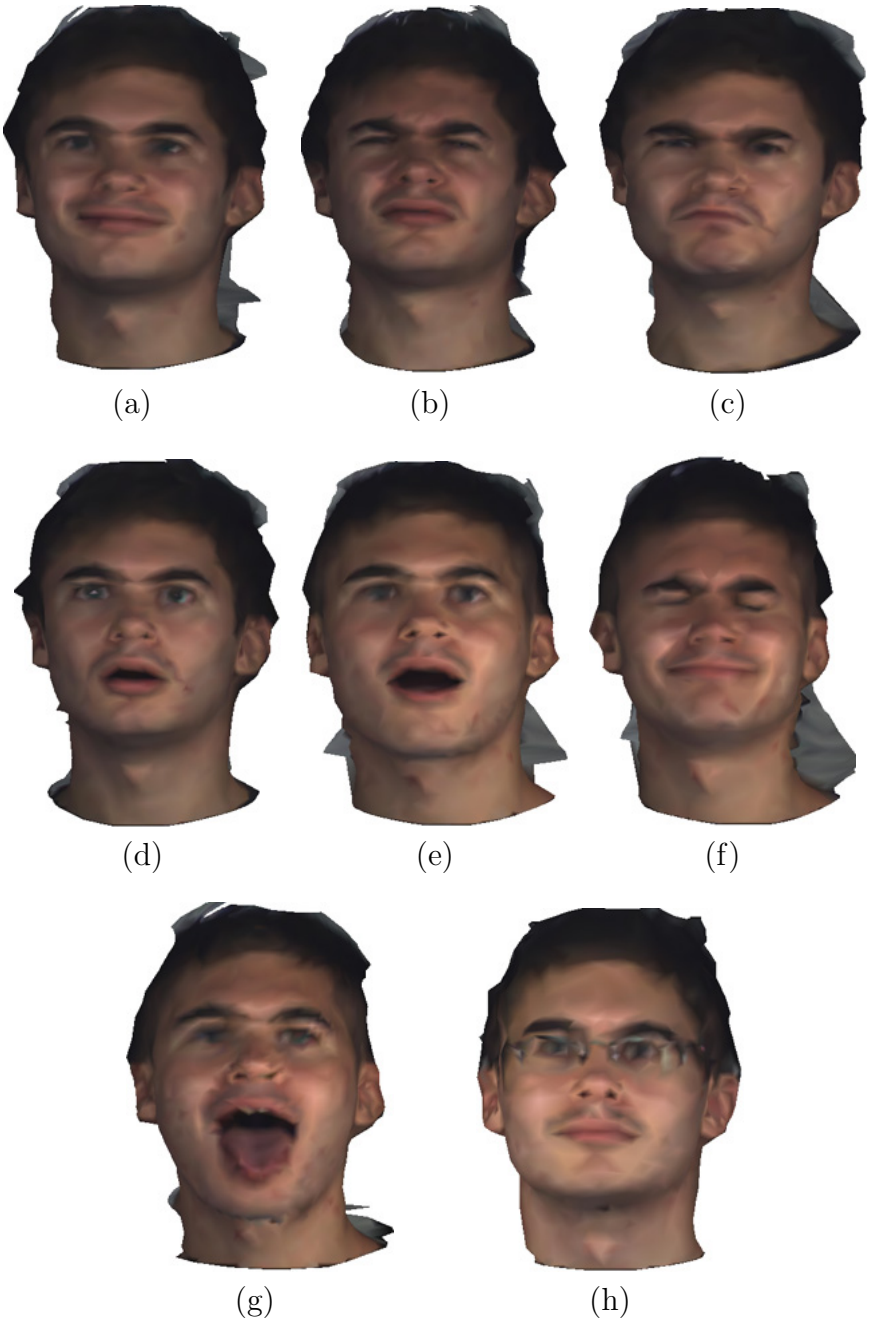


Figure 5.3: **Various facial expressions of a given subject:** (a) "happiness", (b) "disgust", (c) "anger", (d) "fear", (e) "surprise", (f) "eyes shut", (g) "showing his tongue", (h) "wearing glasses". (a) and (b) were used to create the database whereas the other meshes served as test samples. (e)(f)(g) were captured several months after figures (a)(b)(c)(d). Only (g) was not correctly classified in our experiments.

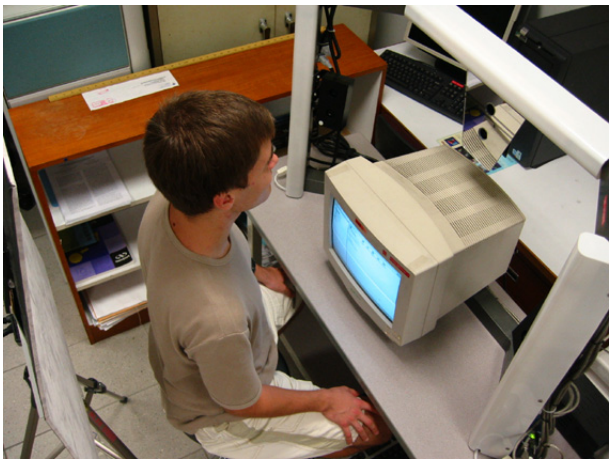


Figure 5.4: Capturing 3D faces with a stereo digitizer. The stereo digitizer from geometrix consists of six high resolution cameras surrounding the subject. A grey back plane is necessary for the reconstruction algorithm to extract the faces from the background.

explained in chapter 2) and it is not accurate when the resolution is low. Our technique remains accurate even at low resolutions.

In the next subsection, we present results for the impostor detection phase described in section 5.1. The overall recognition rates are presented in section 5.3.3 and compared with other classification schemes.

5.3.2 Impostor Detection Results

We created an IMM with the faces of 10 subjects i.e., 20 meshes. We tested our impostor detection algorithm (described in 5.1) with a dataset of 62 faces which included 26 faces of impostors (the remaining 9 subjects) and we obtained **88.77%** correct classifcation. This phase (described in 5.1) does not rely on the specificity of the IMM which integrates different submodels (needed for identification), but rather on the accuracy of the computed correspondences (see section 3.4.2). This result confirms that the computed correspondences are sufficiently accurate. Fig-

ures 5.5 and 5.6 show the approximations yielded by the IMM when the input belongs to the model and when it is an impostor, respectively.

As mentioned in chapter 2 there are very few works in 3D face recognition to allow for a good comparison with our method. Most other works do not discuss their results for impostor detection: either because their dataset does not contain impostors (they study only the identification problem) or an overall recognition rate (which includes identification) is presented. We compare the recognition performance of our algorithm with other works in the following subsection.

However we compare the impostor detection results obtained with our IMM (created with our symmetric matching scheme) and those obtained with Shelton's morphable models [27]. We also created a model with only one face per person (with Shelton's algorithm), to verify the need for at least two faces in the database.

Symmetric Matching	Shelton	
	1 example	2 examples
88.71%	77.42%	82.26%

Table 5.2: Comparison of classification rates for the detection of impostors. We compare the performance of our impostor detection algorithm used with an IMM computed with our method and morphable models computed with Shelton's algorithm for 1 and 2 examples per person in the morphable model.

Table 5.2 presents the impostor detection rates i.e., the percentages of faces that were correctly classified. Clearly, using only one face per example is not enough to achieve high recognition. Our method using symmetric matching to compute the IMM appears to be more accurate than Shelton's original algorithm, which confirms the qualitative results of section 3.4.3. These results can possibly be impoved by using more sophisticated distance functions (for instance distances

between exctracted features) and making the correspondence computations more accurate.

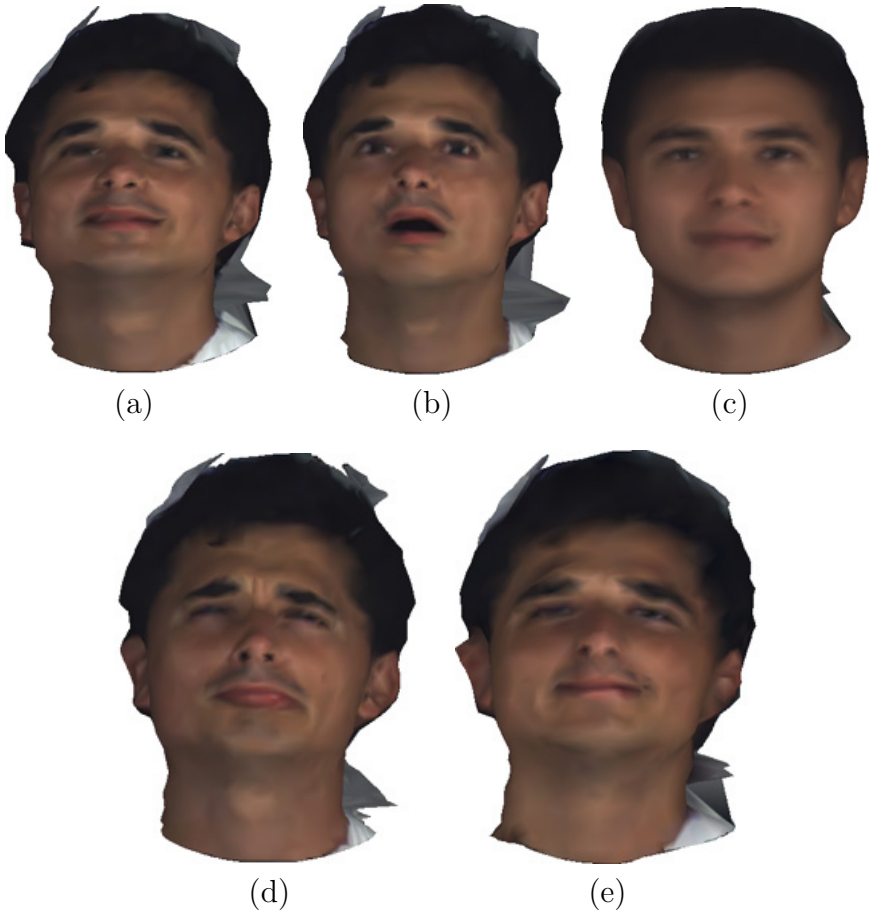


Figure 5.5: **Recognizing whether an arbitrary input mesh belongs to the model.** Figures (a) ("hapiness") and (b) ("surprise") were used to create an IMM (along with faces of 9 individuals). Figure (c) is its base mesh. Figure (e) shows the approximation yielded by the model when matching the arbitrary input mesh, (e) ("sadness").

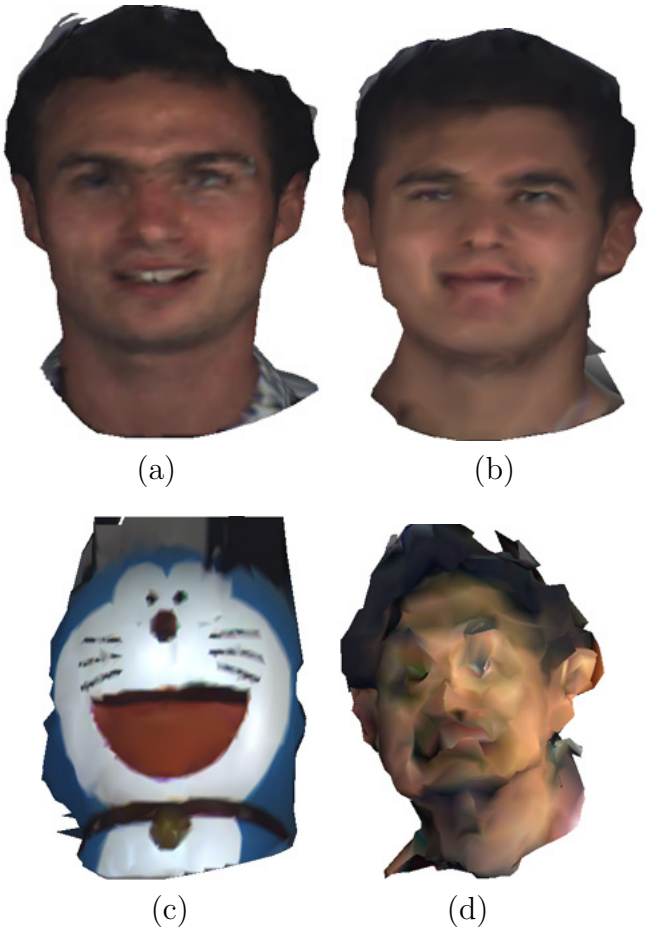


Figure 5.6: **Detecting impostors.** When the model does not contain an example mesh of a subject (figures (a) and (c), an impostor and a toy), it is unable to yield an accurate approximation (figures (b) and (d)). The approximation is even more distorted when the input is not a human face (figure (d)).

5.3.3 Identification Results

In this section, we present results for the identification phase using IMMs described in section 5.2.2 (no impostors in the dataset), and for the overall recognition (i.e., identification with a dataset containing impostors).

An IMM was created with the faces of 13 subjects i.e, 26 meshes. Our algorithm was then tested on the remaining 56 faces of the 13 subjects. We obtained an identification rate of **90.1%** using the reduced morphing vector $\vec{\alpha}'$ and **76.5%** using $\vec{\alpha}$ (unreduced). This confirms the need for reducing the morphing vector as explained in section 5.2.1. The faces that were not identified properly usually exhibited very large expression variations (like the mesh of figure 5.3(g) where the subject has his mouth wide open and shows his tongue). Another type of problem for the algorithm was the difference in hair styles. A female subject had a particular hair style in the meshes used to create the IMM and completely different hair style in the test meshes. For this case only, the test meshes could not be identified.

When the same IMM was tested on a dataset containing impostors (6 impostors forming 7 meshes), our algorithm achieved a very good recognition rate of **92.9%**.

Table 5.3 compares the performance of our algorithm with other works. Our system outperforms similar 3D systems tested under varying expressions. Chang et al. reported an identification rate of 55% in [7] using a Eigenface-based classifier (a performance drop from over 90% when there is no expression change [6]). Moreno et al. reported identification rates from 45% to 62% for a classifier based on feature-extraction and curvature segmentation, the exact figure depending on the expression considered (the best results are obtained for a simple smiling expression, which incurs minimum distortion).

However this is not a fair comparison since these algorithms were not tested on

IMM		PCA-based	Curvature-based
$\vec{\alpha}'$	using $\vec{\alpha}$	Chang [6]	Moreno [19]
93.75%	76.47%	55%	45% to 62%

Table 5.3: Comparison of identification rates obtained with our classifier using either directly the morph parameters α or their reduced version α' and results reported for PCA-based classifier by Chang et al., and a curvature-based classifier by Moreno et al [19].

the same database of faces. There is currently no standardized 3D face database (like FERET for 2D face recognition) and we did not implement Chang's or Moreno's algorithm. Nevertheless Chang's and Moreno's works are quite representative of the state of the art for 3D face recognitions algorithms. Most approaches either use curvature properties like Morenao et al. or adapt techniques similar to PCA like Chang et al. (as explained in chapter 2).

Although the datasets of Chang et Moreno are larger (roughly 200 and 400 faces compared to ours: 60 faces), our results are very promising. Our algorithm maintains a high recognition score even when tested with a much larger variety of expressions (frowns, open mouths, squint..), whereas the performance of PCA or curvature-based algorithms drop with as little 3 different type of expressions in their test sets (we use up from 5 to 8).

The reason for the good performance our algorithm lies in our model's ability to deform to fit the input mesh by morphing and produce an approximation that not only match the identity but also the expression, even if a given subject does not show this particular expression in the training set. In short, our model is able to generalize from the examples faces. The comparison is also biased because we present results using 2 examples per person in the dataset, but this is the minimum required number to create an IMM. But yet another difference is that our algorithm

requires no user intervention, whereas it is not clear if Moreno or Chang edited their mesh before recognition.

We have tested our approach for face recognition with two faces per individual in the IMM, but our the IMM creation algorithm (see section 4.2) supports any number of faces per subject. Using a third face per person would probably further improve the results.

Chapter 6

Conclusion

6.1 Results Summary

In this thesis we have presented a complete framework for the recognition of 3D faces. Our method does not require user intervention and is applicable to most general types of 3D models. We were able to recognize faces with high recognition rates on test samples with very large expression variations.

Our technique builds upon the morphable model approach which is extended for 3D faces and is able to cater for large expression changes.

Morphable models require every example face to be set in point correspondence prior to building the model, which is computationally intensive. We describe the use of adapted data structures to reduce the complexity of the correspondence computation. We introduce a novel symmetric scheme to make the surface correspondences more accurate. Experimental results, qualitative (by visual inspection of the morphing) as well as quantitative (recognition rates), confirm our approach.

We describe a refinement of the morphable model framework to account for expression as well as identity variations in a dataset, with Integrated Morphable Models (IMM). IMMs are created by merging morphable models computed for each person in the database. Recognition can be performed by morphing the model to the input face and carrying out the classification on the morph parameters. We designed a classifier using a reduced version of the morph parameters by keeping only information needed for identification i.e., the information pertaining to expression

variations were discarded.

We carried out experiments on a total of 82 meshes of 19 persons with varying expressions and showed that our classifier outperformed other 3D PCA-based and curvature-based classifiers.

Experimental results confirm the need for using at least two faces per person for higher rates and our model is flexible enough to be used by any arbitrary number of faces per person in the database.

Despite progress in the 3D reconstruction algorithms, artifacts still occur especially at the periphery of meshes captured with 3D stereo digitizers. We presented a method for effectively removing those reconstruction errors.

We also obtained good results for expression synthesis, which emphasizes our model's ability to handle expression variations for a given person or *expressiveness*.

6.2 Future Work

We use a total of 82 different faces for our experiments. There is certainly a need to collect many more to test the system on a larger scale.

The quality of the computed correspondence fields is crucial to obtain high recognition results, as discussed in section 5.1. Therefore improvements in the matching scheme used for computing the correspondences could ultimately lead to better recognition rates. For instance weights could be assigned to points of the surface while matching the mesh, in order to give priority to the most "meaningful" parts of the faces (features such as the nose or the eyes which are more useful for recognition than the hair which is a lot more variable). This could be achieved by introducing weights in the definition of the energy function (see equation 3.2), the

points belonging to such "meaningful" features having the highest weights so that their final correspondences are more accurate.

A preliminary investigation of the prioritized matching scheme gave poor results because of this method's heavy reliance on an accurate segmentation of the features. We believe that if such segmentation could be designed, the prioritized matching could bring significant accuracy improvements to our framework.

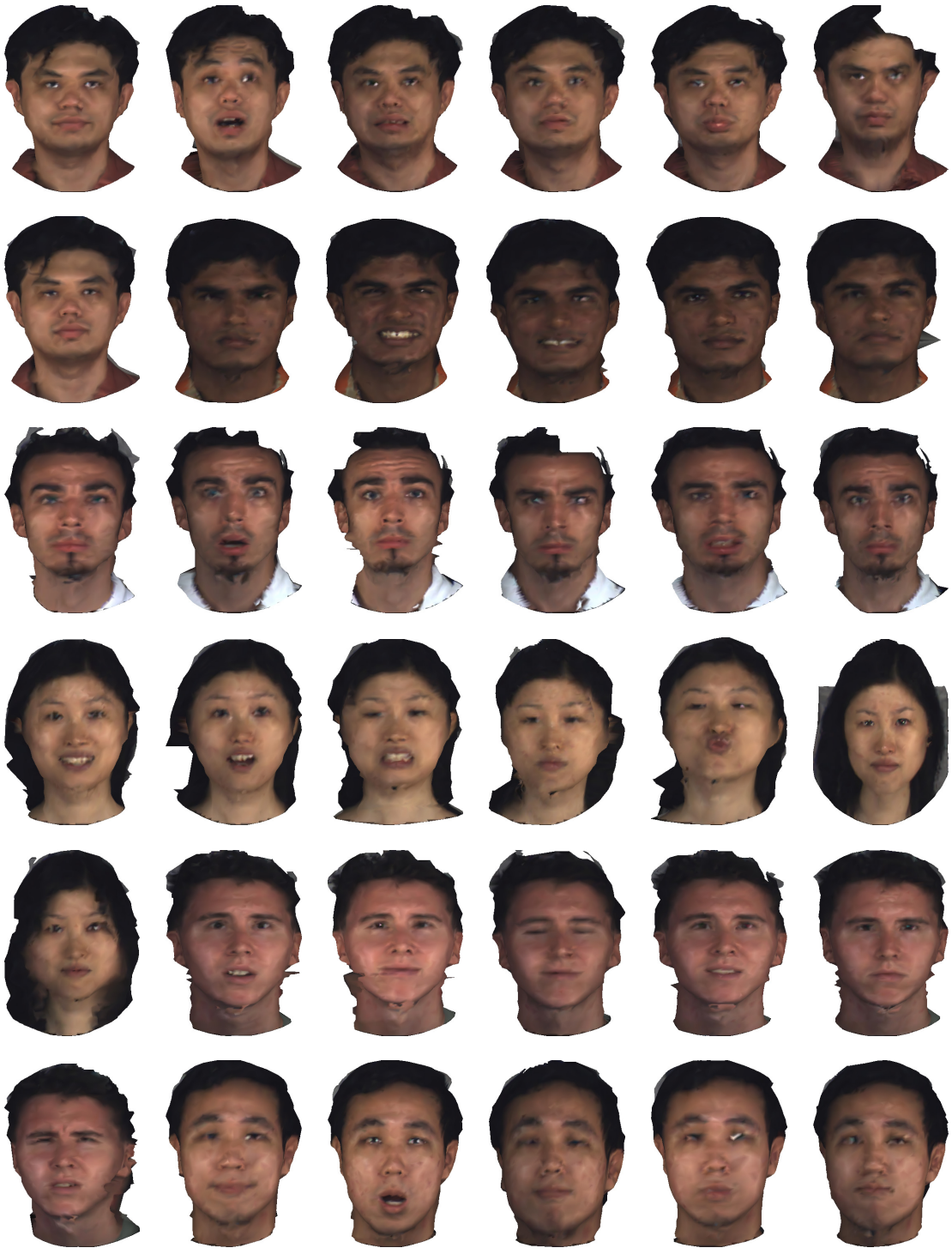
Another possible extension to our work is the recognition of human expressions or emotions. We have shown our model is capable of synthesizing new expressions for a given face, thus our model could be adapted to classify the expression of the input mesh instead of attempting to identity it. But expression recognition requires tight control of expressions displayed by test subjects. The recognition framework would remain the same but the IMM's structure should be modified to cater for expression recognition. The submodels could be formed by matching together faces with the same expression instead of faces having the same identity.

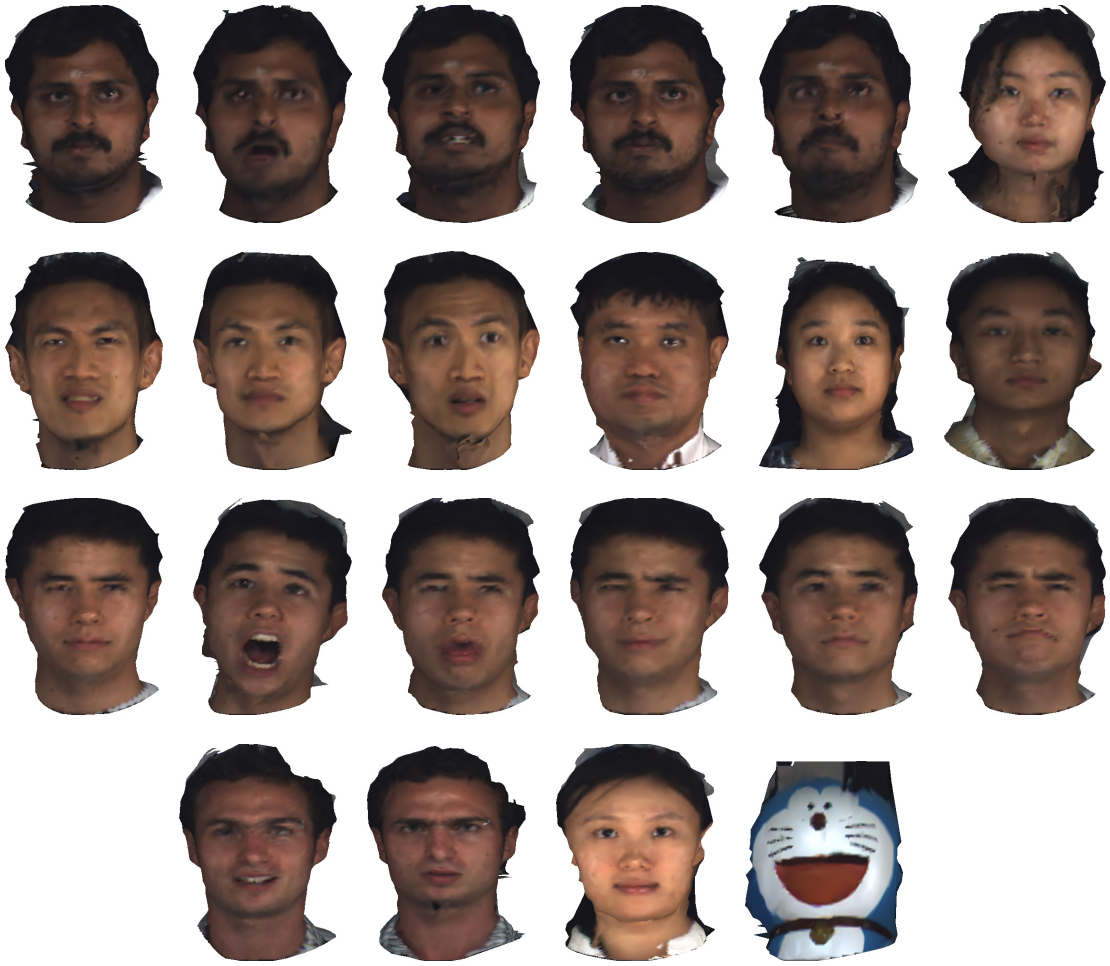
While we have not studied the effects of pose and illumination in our system, our model could be extended to handle large pose or illumination changes as described in previous work for 2D face recognition (see [5]). In this work, a Support Vectors Machines network is trained with examples of faces with different pose and lighting conditions and use the network's generalization capability to classify faces. This technique could be combined with our current algorithm. Other methods to recover the pose of a 3D face (as in [28]) could also be combined with our algorithm.

Appendix A

Database of 3D Faces

In this appendix, the database of 3D faces used in this work (82 faces of 19 subjects) is presented.





BIBLIOGRAPHY

[1] Facevision 600 series , geometrix inc.

[2] B. Achermann and H. Bunke. Classifying range images of human faces with hausdorff distance. In *15th International Conference on Pattern Recognition*, pages 809–813, 2000.

[3] C. Beumier and M. Acheroy. Automatic face authentication from 3d surface, 1998.

[4] D. Beymer and T. Poggio. Face recognition from one example view. In *MIT AI Memo*, 1995.

[5] V. Blanz and T. Vetter. Face recognition based on fitting a 3d morphable model. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(9):1063–1074, 2003.

[6] K. W. Bowyer, K. Chang, and P. J. Flynn. Face recognition using 2d and 3d facial data. In *Workshop on Multimodal User Authentication (MMUA)*, pages 25–32, December 2003.

[7] K. W. Bowyer, K. Chang, and P. J. Flynn. A survey of 3d and multi-modal 3d+2d face recognition. Technical report, Notre Dame Department of Computer Science and Engineering, 2004.

[8] K. W. Bowyer, K. Chang, and P. J. Flynn. A survey of approaches to three dimensional face recognition. In *International Conference on Pattern Recognition*, 2004.

[9] A. Bronstein, M. Bronstein, and R. Kimmel. Expression-invariant 3d face recognition, 2003.

[10] C.-S. Chua, F. Han, and Y.-K. Ho. 3d human face recognition using point signature. In *4th IEEE Int. Conf. on Automatic Face and Gesture Recognition*, pages 233–238, 2000.

[11] P. Ekman. *Emotion in the human Face*. Cambridge University Press, 1982.

[12] G. Gordon. Face recognition based on depth maps and surface curvature. In *Geometric Methods in Computer Vision*, pages 234–247. SPIE Press, 1991.

[13] C. Hesher, A. Srivastava, and G. Erlebacher. A novel technique for face recognition using range images. In *Seventh Intertantional Symposium on Signal Processing and it Applications*, 2003.

[14] H. Hoppe. Progressive meshes. *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 99–108, 1996.

[15] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *SIGGRAPH '93 Proceedings*, pages 19–26, August 1993.

[16] J. Huang, V. Blanz, and B. Heisele. Face recognition using component-based svm classification and morphable models. In *Proceedings of Pattern Recognition with Support Vector Machines, First International Workshop, SVM 2002*, pages 334–341, Niagara Falls, Canada, 2002. Springer 238.

[17] A. P. Mangan and R. T. Whitaker. Partitioning 3d surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, 1999.

[18] M. Meyer. *Discrete Differential Operators for Computer Graphics*. PhD thesis, California Institute of Technology, 2004.

[19] A. Morenoa, A. Sánchez, J. Vélez, and F. Díaz. Face recognition using 3d surface-extracted descriptors. In *Irish Machine Vision and Image Processing Conference (IMVIP 2003)*, 2003.

[20] A. Naftel and Z. Mao. Acquiring dense 3d facial models using structured-light assisted stereo correspondence. Technical report, Computation Department, UMIST, 2002.

[21] J. Oliensis. A critique of structure-from-motion algorithms. *Computer Vision and Image Understanding: CVIU*, 80(2):172–214, 2000.

[22] M. Pantic and Rozenkrantz. Automatic analysis of facial expressions: the state of the art. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 22(12), pages 1424–1445, 2000.

[23] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[24] H. Samet. *The design and analysis of spatial data structures*. Addison-Wesley Longman Publishing Co., Inc., 1990.

[25] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, 2001.

[26] C. R. Shelton. Three-dimensional correspondence. Master's thesis, Massachusetts Institute of Technology, 1998.

[27] C. R. Shelton. Morphable surface models. *International Journal of Computer Vision*, 38(1):75–91, 2000.

[28] L. Shihong, Y. Sumi, M. Kawade, and F. Tomita. Building 3d facial models and detecting face pose in 3d space. *Pattern Recognition Letters*, pages 1191–1202, 2002.

[29] H. T. Tanaka, M. Ikeda, , and H. Chiaki. Curvature-based face surface recognition using spherical correlation and principal directions for curved object recognition. In *3rd IEEE Int. Conf. on Automatic Face and Gesture Recognition*, pages 372–377, 1998.

[30] V.Blanz and T.Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH'99 Conference Proceedings*, pages 187–194, Los Angeles, 1999. ACM Press.

[31] T. Vetter, M. Jones, and T. Poggio. A bootstrapping algorithm for learning linear models of object classes. In *IEEE Computer Vision and Pattern Recognition*, pages 40–46, 1997.

[32] T. Vetter and T. Poggio. Linear object classes and image synthesis from a single example image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):733–741, 1997.

[33] G. Zachmann and E. Langetepe. Geometric data structures for computer graphics. In *Proc. of ACM SIGGRAPH*. ACM Transactions of Graphics, 27–31July 2003.

[34] W. Zhao, R. Chellappa, A. Rosenfeld, and P. Phillip. Face recognition: A literature survey. Technical report, Center for Automation Research, University of Maryland, 2000.